상세보기

(54) COMPRESSION OF TIME-DEPENDENT GEOMETRY

• (19) 국가 (Country) : WO (World Intellectual Property Organization)

• (11) 공개번호 (Publication Number) : WO99/064944 (2000.03.23 Bulletin 2000/13)(See also : WO99/064944 A2)

• (13) 문헌종류 (Kind of Document) : A3 (SUBSEQUENT PUBL. OF THE INT. SEARCH REPORT)
문헌종류코드보기

• (21) 출원번호 (Application Number) : PCT/US99/12732 (1999.06.07)

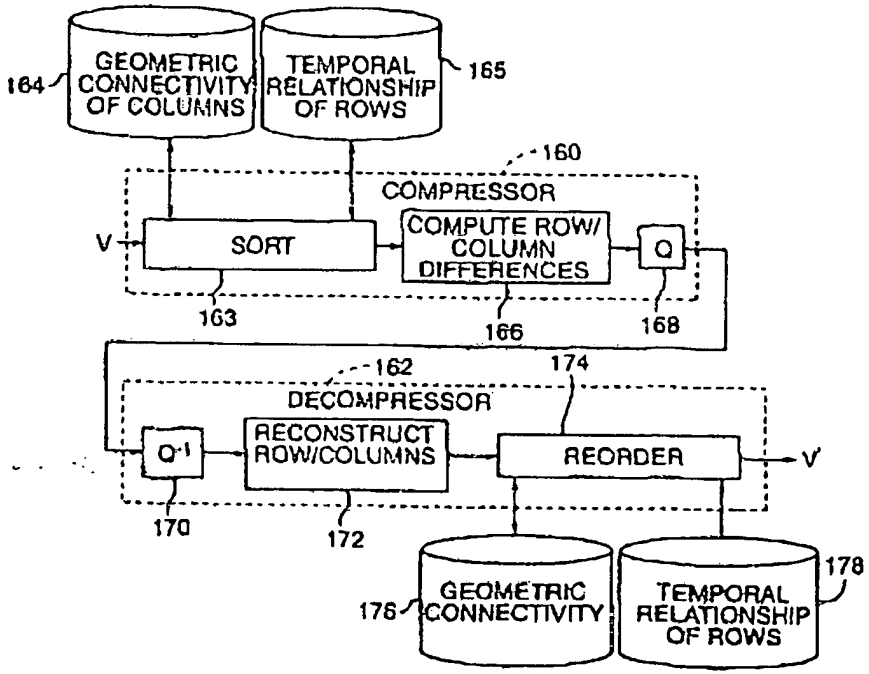
• (75) 발명자 (Inventor) : LENGYEL, Jerome, E.
GUENTER, Brian
MALVAR, Henrique, Sarmento

• (73) 출원인 (Assignee) : MICROSOFT CORPORATION

대표출원인명 : MICROSOFT CORPORATION (A00455)

• (57) 요약 (Abstract) : Methods for coding a time-dependent geometry stream (164, 165) include geometric transform coder, a basis decomposition coder, a column/row prediction coder, and space-time level of detail coder. The basis decomposition coder uses principal component analysis to decompose a time dependent geometry matrix into basis vectors and weights. The weights and basis vectors are coded separately. Optionally, the residual between a mesh constructed from the weight and basis vectors and the original mesh can be encoded as well. The column/row predictor exploits coherence in a matrix of time dependent geometry by encoding differences among neighboring rows and columns (166). Row and column sorting (163) optimizes this form of coding by re-arranging rows and columns to improve similarity among neighboring rows/columns. The space-time level of detail coder converts a time-dependent geometry stream into a hierarchical structure, including levels of detail in the space and time dimensions, and expansion records. The expansion records specify how to reconstruct (172) a mesh from deltas representing differences between levels of detail.
[Show in French]

- 대표도면 :



- (74) 대리인 (Representative) : MEYER, Joel, R.
- (51) 국제특허분류 (IPC 6판) : G06T-017/00
- (30) 우선권번호 (Priority Number) : US 1998-088495 (1998.06.08)
  US 1999-131437 (1999.04.26)
- 본 특허를 우선권으로 한 특허 : AU 4424899 A1 (1999.12.30)
- (81) 지정국 (Designated State) : NATIONAL AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG UZ VN YU ZA ZW GH GM KE LS MW SD SL SZ UG ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG

• 법적상태 (Legal Status) :

Gazette Date	Code	Description (remarks)
1999.06.07	AE	APPLICATION DATA : WO PCT/US99/12732 A (1999.06.07)
1999.12.16	AK	DESIGNATED STATES CITED IN A PUBLISHED APPLICATION WITHOUT SEARCH REPORT(AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG UZ VN YU ZA ZW) : A2
1999.12.16	AL	DESIGNATED COUNTRIES FOR REGIONAL PATENTS CITED IN A PUBLISHED APPLICATION WITHOUT SEARCH REPORT(GH GM KE LS MW SD SL SZ UG ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG) : A2
1999.12.16	A2	PUBLICATION OF THE INTERNATIONAL APPLICATION WITHOUT THE INTERNATIONAL SEARCH REPORT
2000.02.09	121	EP: THE EPO HAS BEEN INFORMED BY WIPO THAT EP WAS DESIGNATED IN THIS APPLICATION
2000.03.23	AK	DESIGNATED STATES CITED IN A SUBSEQUENTLY PUBLISHED SEARCH REPORT (AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG UZ VN YU ZA ZW) : A3
2000.03.23	AL	DESIGNATED COUNTRIES FOR REGIONAL PATENTS CITED IN A SUBSEQUENTLY PUBLISHED SEARCH REPORT(GH GM KE LS MW SD SL SZ UG ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG) : A3
2000.03.23	A3	SUBSEQUENT PUBLICATION OF THE INTERNATIONAL SEARCH REPORT
2000.06.08	DFPE	REQUEST FOR PRELIMINARY EXAMINATION FILED PRIOR TO EXPIRATION OF 19TH MONTH FROM PRIORITY DATE
		ENTRY INTO THE NATIONAL PHASE IN

본 특허의 출원인이 출원한 동일기술문아특허기 없습니다.

2000.12.08	ENP	: JP 2000-553981 A
2001.01.08	NENP	NON-ENTRY INTO THE NATIONAL PHASE IN: : RU
2001.01.08	NENP	NON-ENTRY INTO THE NATIONAL PHASE IN:(RU)
2001.04.12	REG	REFERENCE TO NATIONAL CODE(DE8642) : DE (8642. .)

Code Color : Blue = Positive action, Red = Negative action, Black = Neutral action

- WIPS 패밀리 (※ WIPS 패밀리 보기)

패밀리/법적상태 일괄보기

Up

(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号
特表2002－517851
（Ｐ2002－517851Ａ）
(43)公表日　平成14年６月18日（2002.6.18）

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｔ　15/70		Ｇ０６Ｔ　15/70	Ａ	５Ｂ０５０
17/00		17/00		５Ｂ０８０
Ｈ０４Ｎ　7/24		Ｈ０４Ｎ　7/13	Ｚ	５Ｃ０５９

審査請求　未請求　予備審査請求　有　（全101頁）

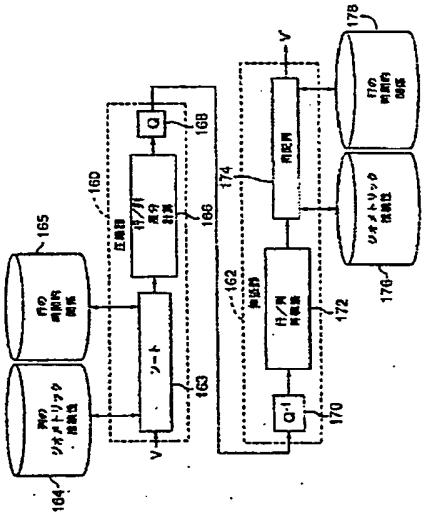
(21)出願番号　特願2000－553881（Ｐ2000－553881）
(86)(22)出願日　平成11年６月７日（1999.6.7）
(85)翻訳文提出日　平成12年12月８日（2000.12.8）
(86)国際出願番号　ＰＣＴ／ＵＳ９９／１２７３２
(87)国際公開番号　ＷＯ９９／６４９４４
(87)国際公開日　平成11年12月16日（1999.12.16）
(31)優先権主張番号　６０／０８８，４９５
(32)優先日　平成10年６月８日（1998.6.8）
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　６０／１３１，４３７
(32)優先日　平成11年４月26日（1999.4.26）
(33)優先権主張国　米国（ＵＳ）

(71)出願人　マイクロソフト　コーポレイション
ＭＩＣＲＯＳＯＦＴ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　ワシントン州　98052－6399　レッドモンド　ワン　マイクロソフト　ウェイ　（番地なし）
(72)発明者　ジェローム　イー．　レンジェル
アメリカ合衆国　98108　ワシントン州　シアトル　ノース　42　ストリート　1808
(74)代理人　弁理士　谷　義一　（外２名）

最終頁に続く

(54)【発明の名称】　時間依存ジオメトリの圧縮

(57)【要約】
時間依存ジオメトリストリームをコード化する方法が開示されている。この方法によれば、ジオメトリック変換コーダ、基底分解コーダ、列/行予測コーダ、および空間的－時間的詳細レベルコーダが含まれている。ジオメトリック変換コーダは、時間依存ジオメトリストリームに含まれる一連のメッシュの各々ごとに、変換ベースメッシュとオリジナルメッシュとの差分を符号化する。ベースメッシュをカレントメッシュに合致させるジオメトリック変換は入力ストリームから導き出すことも、モデルの作成者が与えることもできる。基底分解コーダは、主要コンポーネント分析を使用して、時間依存ジオメトリマトリックスを基底ベクトルと重みに分解する。重みと基底ベクトルは別々にコード化される。オプションとして、重みと基底ベクトルから構築されたメッシュとオリジナルメッシュとの差分を符号化することも可能である。列/行予測器は、隣接行と列間の差分を符号化することで時間依存ジオメトリのマトリックスにおけるコヒーレンスを利用する。行と列をソートすると、行と列が隣接行と列間の類似性を向上するように並べ替えられる



【特許請求の範囲】

【請求項１】 アニメーションで時間依存ジオメトリを表している一連の3Dメッシュを圧縮する方法であって、該方法は、

アニメーションシーケンス内のカレントメッシュの位置に近似化するようにベースジオメトリックメッシュを変換するジオメトリック変換を符号化し、

変換されたベースメッシュの要素とカレントメッシュの対応する要素との間の差分として残余を決定し、

残余、ベースメッシュおよびジオメトリック変換を符号化することを含むことを特徴とする方法。

【請求項２】 請求項１に記載の方法において、さらに、

アニメーションシーケンス内のタイムサンプルごとに、そのタイムサンプルのカレントメッシュの位置に近似化する変換ベースメッシュにベースメッシュを変換するタイムサンプルであって、当該タイムサンプルのジオメトリック変換を見つけることによってジオメトリック変換を決定することを含むことを特徴とする方法。

【請求項３】 請求項１に記載の方法において、さらに、

ジオメトリック変換を量子化し、

ジオメトリック変換を量子化解除し、

ベースメッシュを量子化し、

ベースメッシュを量子化解除し、

量子化解除ベースメッシュを量子化解除変換で変換して変換ベースメッシュを計算することを含むことを特徴とする方法。

【請求項４】 請求項２に記載の方法において、各タイムサンプルのベースメッシュは、残余と先行タイムサンプルの変換ベースメッシュから再構築されたメッシュであることを特徴とする方法。

【請求項５】 請求項１に記載の方法において、さらに残余を表すマトリックスの行または列で行または列予測を使用して残余を符号化することを含むことを特徴とする方法。

【請求項６】 請求項２に記載の方法において、さらに、行または列予測を

使用して残余を符号化することを含むことを特徴とする方法。

【請求項7】 請求項1に記載の方法において、さらに、残余を表すマトリックスを基底ベクトルと重みに分解することを含むことを特徴とする方法。

【請求項8】 請求項7に記載の方法において、重みで行または列予測を実行することを含むことを特徴とする方法。

【請求項9】 請求項1に記載のステップを実行する命令を有するコンピュータ可読媒体。

【請求項10】 時間依存ジオメトリを圧縮する方法であって、

アニメーションシーケンス内のカレントメッシュの位置に近似化するようにベースジオメトリックメッシュを変換するジオメトリック変換を符号化し、

変換されたカレントメッシュとベースメッシュとの間の残余を決定し、

残余、ベースメッシュおよびジオメトリック変換を符号化することからなることを特徴とする方法。

【請求項11】 請求項10に記載の方法において、さらに、

ジオメトリック変換を、カレントタイムサンプルのモデリングおよびカメラ変換と結合することを含むことを特徴とする方法。

【請求項12】 3D時間依存ジオメトリの圧縮ストリームを復号化する方法であって、

アニメーションシーケンス内のオブジェクトのジオメトリを表しているベースメッシュを復号化し、

アニメーションシーケンスにおける一連のタイムサンプル内のタイムサンプルごとに、ジオメトリック変換パラメータの集合を復号化し、

各タイムサンプルの残余が、対応するジオメトリック変換パラメータの集合によって変換されたベースメッシュと、そのタイムサンプルのオリジナルメッシュとの間の差分を表す、各タイムサンプルの符号化残余を復号化し、

各タイムサンプルのオリジナルメッシュを、タイムサンプルに対応する復号化ジオメトリック変換パラメータの集合をベースメッシュに適用することによって再構築することを含むことを特徴とする方法。

【請求項13】 請求項12に記載の方法において、ジオメトリック変換パ

ラメータの集合を復号化するステップは、ジオメトリック変換パラメータを量子化解除することを含むことを特徴とする方法。

【請求項１４】 請求項１２に記載の方法において、ベースメッシュと符号化残余を復号化するステップは、ベースメッシュと符号化残余を量子化解除することを含むことを特徴とする方法。

【請求項１５】 請求項１２に記載の方法において、タイムサンプルごとにオリジナルメッシュを再構築するために使用されるベースメッシュは、先行タイムサンプルの再構築メッシュであることを特徴とする方法。

【請求項１６】 請求項１２に記載の方法において、符号化残余を復号化するステップは、基底ベクトルマトリックスと重みマトリックスから残余マトリックスを再構築することを含むことを特徴とする方法。

【請求項１７】 請求項１２に記載の方法において、符号化残余を復号化するステップは、残余マトリックスの行または列で逆予測を実行することを含み、残余マトリックスの行または列は隣接行または列の対応する要素の間の異なる値から再構築されることを特徴とする方法。

【請求項１８】 3Dオブジェクトの経時的動きを表している3D位置マトリックスを圧縮する方法であって、該方法は、

3D位置マトリックスを基底ベクトルと重みに分解し、

基底ベクトルを符号化し、

重みを符号化することを含むことを特徴とする方法。

【請求項１９】 請求項１８に記載の方法において、3D位置マトリックスは頂点位置マトリックスを含むことを特徴とする方法。

【請求項２０】 請求項１８に記載の方法において、基底ベクトルと重みを符号化することは、基底ベクトルと重みをそれぞれ量子化することを含むことを特徴とする方法。

【請求項２１】 請求項１８に記載の方法において、重みを符号化することは、重みで行または列予測を実行することを含むことを特徴とする方法。

【請求項２２】 請求項２１に記載の方法において、さらに、各行/列内と、対応するレファレンス行/列内の要素の類似性を向上するように列または行を

ソートすることを含むことを特徴とする方法。

【請求項２３】　請求項１８に記載の方法において、さらに、

残余を計算することを含み、その計算は、ジオメトリの時間依存マトリックスを重みと基底ベクトルから再構築し、

再構築されたマトリックスと3D位置マトリックスの対応する要素の間の差分を決定することによって行われることを特徴とする方法。

【請求項２４】　請求項１８に記載の方法において、さらに、頂点位置マトリックスでメッシュ単純化を行ってマトリックスのサイズを縮小してから分解の計算を行うことを含むことを特徴とする方法。

【請求項２５】　請求項１９に記載の方法において、頂点位置マトリックスはアニメーションシーケンスにおける各タイムサンプルの残余を表し、各タイムサンプルの残余は、

ジオメトリック変換パラメータを使用して、タイムサンプルでベースメッシュをオリジナルメッシュに変換し、

タイムサンプルの残余を、変換されたベースメッシュとオリジナルメッシュの対応する要素の間の差分として計算することによって計算されることを特徴とする方法。

【請求項２６】　請求項２５に記載の方法において、ジオメトリック変換パラメータは、タイムサンプルのオリジナルメッシュを、タイムサンプルの変換ベースメッシュに最良に合致させるジオメトリック変換パラメータの集合をタイムサンプルごとに見つけることによって、タイムサンプルごとに求められることを特徴とする方法。

【請求項２７】　請求項１８に記載のステップを実行する命令を有するコンピュータ可読媒体。

【請求項２８】　3Dオブジェクトの経時的動きを表している3D位置マトリックスを伸張する方法であって、該方法は、

基底ベクトルマトリックスを復号化し、

重みのマトリックスを復号化し、

基底ベクトルマトリックスからの3D位置マトリックスと重みのマトリックスを

合成することを含むことを特徴とする方法。

【請求項２９】 請求項２８に記載の方法において、重みのマトリックスを復号化するステップは重みを量子化解除することを含むことを特徴とする方法。

【請求項３０】 請求項２８に記載の方法において、重みのマトリックスを復号化するステップは、重みのマトリックスの行または列で逆予測を実行することを含み、マトリックスの行または列は隣接行または列の対応する要素の間の異なる値から再構築されることを特徴とする方法。

【請求項３１】 請求項２８に記載の方法において、3D位置マトリックスは頂点位置のマトリックスであり、

重みのマトリックスを復号化することは、重みを量子化解除し、重みのマトリックスの列で逆予測を行うことを含み、マトリックスの列は隣接列の対応する要素の間の異なる値から再構築されることを特徴とする方法。

【請求項３２】 請求項２８に記載のステップを実行する命令を有するコンピュータ可読媒体。

【請求項３３】 頂点位置データの時間依存マトリックスを圧縮する方法であって、

隣接行/列の対応する要素の間の類似性を向上するようにマトリックスの行/列をソートし、

隣接行/列の対応する要素の間の異なる値を決定し、

差分値を符号化することを含むことを特徴とする方法。

【請求項３４】 請求項３３に記載の方法において、頂点位置データのマトリックスは重みのマトリックスであり、さらに、

3D頂点位置データのマトリックスを基底ベクトルのマトリックスと重みのマトリックスに分解することを含むことを特徴とする方法。

【請求項３５】 請求項３３に記載の方法において、頂点位置データのマトリックスはタイムサンプルを表す行とジオメトリック位置データを表す列を含んでいることを特徴とする方法。

【請求項３６】 アニメーションにおいて時間依存ジオメトリを表している一連の3Dメッシュを圧縮する方法であって、該方法は、

頂点位置のマトリックスを、ジオメトリック詳細レベルと時間的詳細レベルを含む空間と時間の階層として時間依存メッシュを表している階層データ構造に変換し、

各詳細レベル間のデルタを符号化し、

ジオメトリックまたは時間的詳細レベルをどのように拡張して、アニメーションシーケンス内のタイムサンプルのデルタからメッシュを再構築するかを指定する拡張レコードを符号化することを含むことを特徴とする方法。

【請求項37】 請求項36に記載の方法において、さらに、

タイムサンプルごとに拡張レコードを選択的に送信してタイムサンプルのメッシュを再構築することを含むことを特徴とする方法。

【請求項38】 請求項37に記載の方法において、さらに、

カレントタイムサンプルに対するメッシュのスクリーン空間投影を評価して、カレントタイムサンプルに対してどの拡張レコードを送信するかを決定することを含むことを特徴とする方法。

【請求項39】 請求項36に記載の方法において、頂点位置のマトリックスを階層データ構造に変換することは、

マトリックスを、独立にアニメーション化されるセグメントに分割し、各セグメントを階層データ構造に変換することを含むことを特徴とする方法。

【請求項40】 請求項36に記載の方法において、ジオメトリック詳細レベルは、頂点位置を相互接続するエッジがより少ない頂点のメッシュを作成するように縮小されているメッシュエッジ縮小（mesh edge collapse）を使用してコード化されるか、または

メッシュ頂点位置を追加の頂点位置に分割し、より詳細化されたメッシュを作成することよってコード化されることを特徴とする方法。

【請求項41】 請求項36に記載の方法において、頂点位置のマトリックスを階層データ構造に変換することは、

ベースメッシュ内のジオメトリが変化するジオメトリック詳細レベルで表現されているメッシュ精細化の階層にベースメッシュを変換し、

選択した時間インターバルの各メッシュを、選択した時間インターバルのメッ

シュ内のジオメトリが変化するジオメトリック詳細レベルで表現されているメッシュ精細化の階層に変換することを含むことを特徴とする方法。

【請求項４２】 請求項３６に記載の方法において、時間的詳細レベルは、一連の3Dメッシュ内の２または３以上の時間的詳細レベルの間に補間することによって作られることを特徴とする方法。

【請求項４３】 請求項４２に記載の方法において、時間的補間はスプライン補間であることを特徴とする方法。

【請求項４４】 請求項４３に記載の方法において、一連の3Dメッシュ内の対応する頂点位置は時間を通るようにスプラインされ、より高い時間的詳細レベルは結び目の挿入によって追加されることを特徴とする方法。

【請求項４５】 請求項４３に記載の方法において、一連の3Dメッシュ内の対応する頂点位置は時間を通るようにスプラインされ、より高い時間的詳細レベルはウェーブレットと共に追加されることを特徴とする方法。

【請求項４６】 請求項３６に記載の方法を実行する命令を有するコンピュータ可読媒体。

【請求項４７】 頂点位置の時間変化マトリックスをジオメトリックおよび時間的詳細レベルの階層として表現しているデータ構造を使用して、3Dアニメーションにおける時間依存ジオメトリを選択的にプレイバックする方法であって、該方法は、

クライアントからサーバに対して、アニメーションのカレントビューとジオメトリック詳細レベルを指定し、

クライアントでは、サーバからの、ジオメトリック詳細レベルを示していて、アニメーション内で出力画像にレンダリングするために階層データ構造から抽出される空間的更新レコードを受信し、

クライアントからサーバに対して、時間的詳細レベルとカレント時間インターバルを指定し、

クライアントでは、サーバからの、時間的詳細レベルを示していて、時間インターバルでのアニメーション内で出力画像にレンダリングするために階層データ構造から抽出される時間的更新レコードをクライアントで受信することを含むこ

とを特徴とする方法。

【請求項４８】 請求項４７に記載の方法において、空間的更新レコードは、頂点位置マトリックスのジオメトリック詳細レベルを、それぞれ粗粒化または精細化するために使用される粗粒化または精細化レコードであることを特徴とする方法。

【請求項４９】 請求項４７に記載の方法において、アニメーション内のオブジェクトにマッピングされるテクスチャ画像の空間的解像度は、テクスチャが変化する空間的解像度で表現され、第１詳細レベルが最初にサーバからクライアントに転送され、その後、それより高または低の詳細レベルがクライアントに通知されて、テクスチャ画像の選択した詳細レベルが得られるように階層的に符号化されることを特徴とする方法。

【請求項５０】 請求項４９に記載の方法において、テクスチャはアニメーション化され、テクスチャ画像の時間的詳細レベルは、更新レコードをクライアントに送信することによって選択的に増減されて、選択した詳細レベルを供給することを特徴とする方法。

【請求項５１】 請求項５０に記載の方法において、サーバは、クライアントからの選択した詳細レベルの要求に応答して、テクスチャ詳細レベルの更新をクライアントに送信することを特徴とする方法。

【請求項５２】 請求項１２に記載の方法を実行するソフトウェアを有するコンピュータ可読媒体。

【請求項５３】 アニメーションで移動するジオメトリを表している一連の3Dメッシュを符号化する方法であって、該方法は、

頂点位置のマトリックスを、ジオメトリック詳細レベルと選択した次元が変化する詳細レベルで各々符号化されているジオメトリック詳細レベルと選択した空間的または時間的次元の階層としてメッシュを表している階層データ構造に変換し、

各更新レコードが階層データ構造から抽出される所望の詳細レベルを指定している更新レコードを符号化することを特徴とする方法。

【請求項５４】 請求項５３に記載の方法において、選択した次元は空間的

次元であり、空間的次元内の所望の位置は、空間的次元に沿うアニメーションのプレイバックを制御するためにユーザ入力を通して指定されることを特徴とする方法。

【請求項５５】　請求項５３に記載の方法において、選択した次元は時間的次元であり、所望の時間的次元は、時間的次元に沿うアニメーションのプレイバック期間に時間的詳細レベルを制御するように指定されることを特徴とする方法。

【発明の詳細な説明】

【0001】

発明の分野

本発明はコンピュータ生成グラフィックスに関し、さらに具体的には、時間依存（time dependent）ジオメトリック（幾何図形）データの圧縮に関する。

【0002】

発明の背景

3次元（3D）グラフィックスでは、移動するオブジェクトは3Dジオメトリックモデル（geometric model－幾何モデルまたは図形モデル）を使用してモデリングされている。これらのモデルは、3D空間におけるオブジェクトの表面を表している表面要素（surface element）のメッシュの位置を定義している3D座標の集合として表現されているのが代表的である。3Dオブジェクトモデルを含んでいるシーンをレンダリング（rendering）するために、グラフィックスレンダリングパイプライン(graphics rendering pipeline)は、最初に、一連のジオメトリック変換（geometric transformation－図形変換）を実行してモデルをローカル座標空間からシーンのグローバルまたは「ワールド」座標系に変換してから2Dビュー空間のビューイングまたは「カメラ」座標に変換している。次に、変換されたジオメトリとその属性（カラー、陰影など）を出力イメージ（画像、映像など。以下「画像」という）を表すピクセル値の配列に変換している。このプロセスは、オブジェクトモデルがシーンの周りを移動するとき、アニメーションシーケンスの画像フレームごとに繰り返されているのが代表的である。

【0003】

移動するグラフィカルオブジェクトは時間依存ジオメトリ（time-dependent geometry）の形で表現されている。モデルの3D位置を表しているメッシュは時間の経過と共に移動し、変形して、3Dオブジェクトの動き（モーション）をシミュレートしている。アニメーションで3Dジオメトリの動きを描写するために使用される動きモデル（motion model）にはいくつかがある。比較的単純なオブジェクトは剛体（rigid body）でジオメトリック変換を使用してモデリングすることができる。しかし、計算能力が向上し、より現実的なアニメーションの要求に伴い

、複雑なアニメーションモデルをリアルタイムでプレイバックするようなアプリケーションの要求が高まっている。

【0004】

Avid Technology, Inc. 提供のSoftimageモデリングツールのような、精巧なオーサリングツール（authoring tool）やモデリングプログラムは、非常に複雑な時間依存ジオメトリを作成する能力を備えている。複雑に移動するジオメトリシーメーンスは、フリーフォーム変形格子（free-formed deformation lattice）、ジョイントエンベロープ（joint envelope）や物理的シミュレーション、その他の操作によって作ることが可能になっている。アニメーション変換（animated transformation）による単純な剛体モデル以上のものがリアルタイプアプリケーションに要求されることに伴い、複雑なアニメーションモデルをリアルタイムのパフォーマンスで効率的にストアし、プレイバックする方法を開発することがますます緊急になっている。

【0005】

オーサリングツールがますます精巧化されていることのほかに、3Dキャプチャシステム（capture system）の進歩により、3Dグラフィックスアプリケーションで使用される時間依存ジオメトリもさらに複雑化することが見込まれている。ここで、「3Dキャプチャ」という用語は、現実モデルのデジタル化3Dモデルを生成するプロセスのことを意味している。現在は、スタチック（静的）ジオメトリ集合はレンジスキャナ（range scanner）で得ている。しかし、レンジスキャナの精度とスピードが向上するのに伴い、大きな時間依存ジオメトリックメッシュのソースが増加している。シミュレーションは、リッチなアニメーションジオメトリを得る別のソースになっている。有限要素（definite element）法によると、現実的で複雑なアニメーションが得られるが、リアルタイムで計算するにはコストがかかりすぎる。

【0006】

複雑な時間依存ジオメトリを得るソースが普及するのに伴い、必要なメモリ量と帯域幅を低減化するために時間依存ジオメトリをもっと効率よくストアし、伝達するための方法に対する要求が高まっている。研究者は、スタチックジオメト

リを圧縮する方法を研究している。参考文献として、Michael F. Deering著「ジオメトリック圧縮(Geometric Compression)」(pp. 13-20, SIGGRAPH 95)、Mike M. Chow著「リアルタイムレンダリングのための最適化ジオメトリ圧縮(Optimized Geometry Compression for Real-Time Rendering)」(pp. 347-354, Proceedings of IEEE Visualization, 97)、Stefan Gumhold、Wolfgang Strasser共著「トライアングルメッシュ接続性のリアルタイム圧縮(Real Time Compression of Triangle Mesh Connectivity)」(pp. 133-140, SIGGRAPH 98)、Gabriel Taubin、Jarek Rossignac共著「トポロジカルサージャリによるジオメトリック圧縮(Geometric Compression Through Topological Surgery)」(ACM Transactions on Graphics, Vol. 17, No. 2, April 1998, pp. 84-115)、Gabriel Taubin、Andre Gueziec、William Horn、Francis Lazarus共著「プログレッシブフォレストスプリット圧縮(Progressive Forest Split Compression)」(pp. 123-132, SIGGRAPH 98)、Costa Touma、Crag Gotsman共著「トライアングルメッシュ圧縮(Triangle Mesh Compression)」(Proceedings of Graphics Interface, ' 98, pp. 26-34)、およびFrank Bossen (編者) 「3Dモデルコーディングでのコア実験の説明(Description of Core Experiments on 3D Mode Coding)」(ISO/IEC JTC1/SC29/WG11 MPEG98/N244rev1, Atlantic City, October 1998)がある。この研究では、スタチックジオメトリの圧縮が対象になっているが、移動する3Dジオメトリストリームを圧縮する方法を開発するには、もっと多くの作業が必要である。

【0007】

3Dジオメトリの圧縮とは対照的に、静止画像と移動画像圧縮の分野は十分に開発されている。静止画像の圧縮のために使用できる手法には、ランレングス符号化 (run-length encoding)、JPEG符号化などのように、さまざまな手法がある。画像シーケンスの圧縮手法も、MPEG、AVIなどのように多数の手法がある。研究者は、3Dジオメトリを使用してムービー (動画) 圧縮を支援する手法も提供している。参考文献としては、Brian K. Guenter、Hee Cheol Yun、Russell M. Mersereau共著「コンピュータアニメーションフレームの動き補償圧縮(Motion Compensated Compression of Computer Animated Frames)」(pp. 297-304, SIGGRAPH ' 93)、Mark Levoy著「合成画像のポリゴン支援JPEGおよびMPEG圧縮(Polygon-As

sisted JPEG and MPEG Compression of Synthetic Images)」(pp. 21-28, SIGGRAPH ' 95)、およびDan S. Wallach、Sharma Kunapalli、Michael F. Cohen共著「ダイナミックポリゴナルシーンの高速MPEG圧縮(Accelerated MPEG Compression of Dynamic Polygonal Scenes)」(pp. 193-197, SIGGRAPH ' 94)がある。

【0008】

ある側面では、従来のグラフィックスレンダリングパイプラインによる手法では、アニメーションオブジェクトが一連のアニメーション変換マトリックスを使用して変換されるスタティックな剛体で表されている場合、アニメーションジオメトリの圧縮が行われている。このケースでは、時間依存ジオメトリックモデルは、剛体を表すシングルメッシュと、時間の経過と共に剛体の動きを記述する一連のアニメーション変換マトリックスにレンダリングされている。このように単純にコヒーレント部分に分割すると、移動するオブジェクトは剛体の階層として構築できるので大きな時間依存アニメーション群の符号化(encode)が可能になっている。この方法によると、限られたクラスの時間依存ジオメトリは効率よく圧縮されるが、もっと複雑なアニメーションモデルをもっと一般化され、フレキシブルな方法で圧縮するという要求には、十分に応えていない。複雑な動きの形態によっては、剛体の階層および関連変換マトリックスの使用によるシミュレーションに適さないものがある。さらに、モデルによっては、剛体から構築されないものがあり、むしろ、これらのモデルは、オーサリングツールや3Dキャプチャツールのように、ジオメトリが剛体の形で表されていないジオメトリソースから得られている。

【0009】

発明の概要

本発明は、時間依存ジオメトリおよびアニメーションをコード化(code)する方法を提供している。本発明の側面は、アニメーション3Dオブジェクトだけでなく、時間以外の次元と共に変化する3Dアニメーションも表している時間依存メッシュのエンコーダ(符号器)およびデコーダ(復号器)で実現されている。これらの手法を使用すると、3Dジオメトリストリームのストアと転送が効率化される。これは、コンピュータシステム内で使用すると、ホストプロセッサまたはスト

レージデバイスとグラフィックスレンダリングエンジン（graphics rendering engine）との間の帯域幅が低減されるので便利である。また、ローカルエリアネットワーク（local area network — LAN）や広域ネットワーク（wide area network — WAN）上のコンピュータ相互間の伝送帯域幅を低減する上でも便利である。さらに、これらの手法は、ジオメトリストリームが生成され、符号化され、その後、即時プレイバックのために復号化されるアプリケーションのように、ジオメトリストリームが時間的拘束の中で符号化されるような、ダイナミック圧縮コンテキストで使用すると便利である。

【００１０】

一般的に、本発明の圧縮方法によれば、ジオメトリストリームは、そのストリームの低パラメータモデルを得るように分解し、その残余を符号化することによってコード化されている。圧縮器（compressor）は、オブジェクトの3D位置を選択したタイムサンプル（selected time samples）で表している時間依存ジオメトリ構造に作用する。具体的には、以下で説明するコーダ（coder）は、アニメーションシーケンス内の一連のタイムサンプルに対するメッシュの3D位置を表している頂点（vertex）位置（マトリックスの列）のマトリックスを圧縮することが中心になっている（マトリックスの行は選択したタイムサンプルでのメッシュを表している）。圧縮器（compressor）は各タイムサンプルのメッシュに近似化し、その近似化メッシュとマトリックスの行からの実際のメッシュとの間の残余（residual）を復号化（decode）する。また、圧縮器は、ジオメトリまたはベースメッシュ（base mesh）のコヒーレント部分、残余、およびメッシュに近似化するために使用されるパラメータを符号化する。伸張器（decompressor）は、圧縮されたジオメトリストリームを復号化し、選択したタイムサンプルのときのメッシュを、コヒーレント部分、残余および各メッシュに近似化するために使用されたパラメータから再構築する。

【００１１】

コーダの一形態として、ジオメトリック変換コーダ（geometric transform coder）、もっと具体的には、メッシュ変換コーダ（mesh transform coder）と呼ばれているものがある。このコーダは、入力マトリックス内の各メッシュと変換さ

れたベースメッシュとの差分（difference）を決定することによって残余を符号化している。ジオメトリック変換は、ベースメッシュと、カレントタイムサンプルのときのメッシュとの間の動きに近似化するために使用されている。ジオメトリック変換はコーダに与えることも、ベースメッシュを各タイムサンプルでのメッシュにマッチング（合致させること）することによって導き出すことも可能である。同様に、ベースメッシュはコーダに与えることも、頂点位置の入力マトリックスから導き出すことも可能である。

【0012】

別形態のコーダとして、列/行予測器（column/row predictor）がある。この形態のコーダは、時間依存ジオメトリを表すマトリックスでコヒーレンスを利用している。例えば、頂点位置のマトリックスの時間的コヒーレンス（temporal coherence）は、マトリックス内の隣接行（neighboring rows）間の差分を符号化することによって利用できる。空間的コヒーレンス（spatial coherence）は、マトリックス内の隣接列（neighboring columns）間の差分を符号化することによって利用できる。コード化効率を向上するには、行と列は、隣接行/列間の差分が最小限になるようにソートすることができる。列と行の予測は頂点位置を表すマトリックスに適用され、この中には、入力マトリックス、残余マトリックス、およびこれらのマトリックスのどちらかを主要コンポーネント分析（principal component analysis）することで得られる重み（weight）マトリックスが含まれている。

【0013】

さらに別形態のコーダとして、基底分解コーダ（basis decomposition coder）がある。主要コンポーネント分析を使用して、時間依存ジオメトリのマトリックスは、基底ベクトルと重みに分解される。基底ベクトルと重みは、オリジナルマトリックスの近似を基底ベクトルと重みから再構築し、その後オリジナルマトリックスと近似マトリックスとの差分を符号化することによって、残余を計算するために使用できる。この形態のコーダでは、残余は基底ベクトルおよび重みと一緒に符号化される。別の基底分解コーダでは、時間依存ジオメトリのマトリックスは基底ベクトルと重みに分解され、その後、重みで列/行予測が使用されて

重みが符号化されている。

【0014】

さらに別形態のコーダでは、時間依存ジオメトリのマトリックスは時間依存ジオメトリを階層空間－時間ピラミッド（space-time pyramid）として表しているデータ構造に変換されている。このデータ構造は、空間と時間詳細レベル(level of detail)と、各詳細レベル間のデルタを符号化している。これが一種の圧縮であるのは、マトリックスのサイズが大きくなっても、デルタが小さいからである。

【0015】

この空間－時間詳細レベルを表現すると、特定の空間－時間構成で必要になる空間的または時間的精細化（spatial or temporal refinements）だけを、エンコーダからデコーダに送ればよいので、送信期間中に別形態の圧縮が得られることになる。圧縮表現全体を送信するのではなく、デコーダは、要求された詳細レベルをフィードバックチャネルでデコーダに送信し、その後、エンコーダは、空間－時間のカレント領域用に要求された詳細レベルを達成するために必要な精細化だけを送信する。

【0016】

空間と時間にわたるアニメーションを符号化するために使用される原理は、他の次元にも応用可能である。例えば、時間次元の代わりに、ジョイントアングル（関節角度）、回転軸、並行移動軸などの、別の次元にわたってジオメトリストリームを符号化することが可能である。このケースでは、ジオメトリは空間次元と新次元（例えば、時間次元の代わりに）内に階層的に符号化される。このように次元に沿って符号化することをインタラクティブアニメーションで利用すると、ユーザはアニメーションオブジェクトの動きを次元に沿って制御できるので便利である。ユーザがこの次元に沿う詳細レベルを指定して入力すると、コーダは該当詳細レベルをジオメトリの階層表現からどのように抽出するかを指定している更新レコードを計算する。

【0017】

上述したコード化手法は、アニメーションで使用される他の形態のジオメトリ

ックデータにまで拡張可能である。例えば、これらの手法はテクスチャ座標（texture coordinate）のコード化に応用されている。

【0018】

以下では、本発明の特徴の理解を容易にするために、添付図面を参照して詳しく説明することにする。

【0019】

詳細説明

序論

アニメーション3Dオブジェクトの動きを記述する時間依存ジオメトリは、次に示す3D位置のマトリックスPで表すことができる。

【0020】

【数1】

$$P = \overbrace{\begin{matrix} p_0^0 & \cdots & p_0^{\alpha_{\omega-1}} \\ \vdots & & \vdots \\ p_{\beta_{\omega-1}}^0 & \cdots & p_{\beta_{\omega-1}}^{\alpha_{\omega-1}} \end{matrix}}^{\text{連続位置}} \Bigg\} \text{連続時間}$$

【0021】

マトリックスPの各列は、3Dメッシュの頂点位置のように、3D位置を表している。行は、アニメーションシーケンスのフレームのように、時間のインクリメント（増分）を表している。マトリックスの各要素のスーパスクリプトは3D位置を示し、この表記では、列番号に対応している。マトリックス内の各要素のサブスクリプトはタイムインクリメントを示し、マトリックスの行番号に対応している。マトリックスPが空間的に連続していると考えると、あるオブジェクトの3D表面を完全に記述する3D位置（列）は、概念的には、無限個数（infinite number）になる。実際には、アニメーションの各タイムインクリメントの3D位置は有限個数（finite number）であり、位置は、その時点でのアニメーションのジオメトリック詳細レベルに対応している。マトリックスPが時間的に連続していると考えると、行は、概念的には、無限個数になる。グラフィックレンダリングパイ

プラインでは、当然のことであるが、3Dモデルは離散的時間で出力画像にレンダリングされ、最も起こり得ることは、フレームレートのように周期的レートで出力画像にレンダリングされている。ある種のアーキテチャでは、オブジェクトの更新レートが変化し得るため、各行はアニメーションシーケンスのフレームに対応している必要がない。もっと一般化された表現では、行はタイムサンプルに対応しているだけである。

【0022】

マトリックスPは、次式に示すように3つのコンポーネントに因数分解（factorize）することができる。

【0023】

【数2】

$$P = \underbrace{\begin{bmatrix} S \end{bmatrix}}_{\text{時間補間}} \underbrace{\begin{bmatrix} V \end{bmatrix}}_{\text{頂点位置}} \underbrace{\begin{bmatrix} G \end{bmatrix}}_{\text{ジオメトリ補間}}$$

【0024】

コンポーネントGは、低レベルのグラフィックスハードウェアに実装されているポリゴン補間（polygon interpolation）または表面パッチ補間（surface patch interpolation）である。コンポーネントSは時間を通るマトリックスの補間である。言い換えれば、時間補間コンポーネントは、3Dモデルが出力画像にレンダリングされるときの離散的時間を指定している。空間的および時間的詳細レベルをジオメトリ補間関数と時間補間関数で制御すると、任意の空間－時間ポリトープ（polytope）はこの公式を使用して符号化することができる。

【0025】

時間とジオメトリ補間コンポーネントはグラフィックスレンダリングハードウェアに実装されているのが代表的であるので、以下に説明する圧縮手法では、頂点位置マトリックスVが中心になっている。

【0026】

頂点位置マトリックスVの各列は、単一頂点の動きを記述している。

【0027】

【数3】

$$V=\overbrace{\left.\begin{bmatrix} v_0^0 & \cdots & v_0^{n-1} \\ \vdots & & \vdots \\ v_{m-1}^0 & \cdots & v_{m-1}^{n-1} \end{bmatrix}\right\}}^{\text{頂点位置}}\ \text{タイムサンプル}$$

【0028】

マトリックスVの列は頂点位置を表している。具体的には、マトリックスVの各要素のスーパスクリプトは頂点であることを示している。マトリックスVの行はタイムサンプルを表している。マトリックスVの各要素のサブスクリプトはタイムサンプルであることを示している。

【0029】

汎用重み付け軌跡 (Generic Weighted Trajectories)

以下に説明する手法はすべて、次のメッシュアニメーション手法を特殊化したものと考えることができる。

【0030】

【数4】

$$V=\begin{bmatrix} \overbrace{\begin{matrix} d_0^0 & \cdots & d_0^{k-1} \\ \vdots & & \vdots \\ d_{M-1}^0 & \cdots & d_{M-1}^{k-1} \end{matrix}}^{\text{軌跡}} & \overbrace{\begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^{N-1} \\ \vdots & & \vdots \\ \alpha_{k-1}^0 & \cdots & \alpha_{k-1}^{N-1} \end{bmatrix}}^{\text{影響}} \end{bmatrix}$$

【0031】

アフィン変換 (Affine Transformations)

代表的なグラフィックスアーキテクチャでは、マトリックスVは、剛体オブジェクトの階層に因数分解されている。これが、上述したもっと一般化されたマトリックスVに比べて小さなクラスの移動ジオメトリであるのは、特定タイプの動き、つまり、剛体で行われる一連のアフィン変換（affine transform）に限定されているためである。この概念を具体例で示すと、剛体動きのマトリックスVの特殊ケースは次式で表すことができる。

【0032】

【数5】

$$V_R = \overbrace{\begin{bmatrix} A_0^0 & \cdots & A_0^{R-1} \\ \vdots & \ddots & \vdots \\ A_{m-1}^0 & \cdots & A_{m-1}^{R-1} \end{bmatrix}}^{\text{アフィン変更}} diag(\overbrace{\hat{V}^0 \cdots \hat{V}^{R-1}}^{\text{剛体}})$$

【0033】

このケースでは、剛体の時間依存ジオメトリックマトリックス$P_R = V_R G$は、3つのコンポーネントに因数分解されている。すなわち、アフィン変換、剛体、およびジオメトリ補間である。左側のマトリックスはアフィン変換を表し、各行はアニメーションのフレームに対応し、各列は剛体のアフィン変換に対応している。上式では、左側マトリックスのアフィン変換が唯一の変化する項であり、フレームごとに更新される。上に示す剛体の各々は剛体構造を構成する頂点の集合からなり、アフィン変換の列が剛体構造と関連付けられている。ジオメトリ補間コンポーネントは各剛体の補間関数からなり、この補間関数は剛体内の頂点にストアされた属性を保管して、デバイス座標を出力するために使用される。

【0034】

グラフィックスレンダリングアーキテクチャには、基本タイプとして即時モード（immediate-mode）アーキテクチャと保存モード（retained-mode）アーキテクチャの2つがある。即時モードアーキテクチャでは、頂点とジオメトリック接続性マトリックス（geometric connectivity matrix）全体がフレームごとに再

送信される。このプロセスによると、3Dジオメトリレンダリングシステムにフレームごとに送信する必要のあるデータが大量であるために、大量の帯域幅が必要になる。これとは対照的に、保存モードアーキテクチャでは、頂点とジオメトリック接続性マトリックスは、あるシーケンスのフレームの間一度だけ送信され、フレームごとに剛体の位置をどのように変更すべきかを3Dレンダリングサブシステムに通知するために使用される変化アフィン変換が符号化される。

【0035】

事実、マトリックスVの列を、コヒーレントな動きをもつ剛体の集合にセグメント化することが1つの圧縮方式となっているのは、この方式によると、3Dジオメトリのフレーム間の (frame to frame) 変化がアフィン変換で符号化できるからである。しかし、この動きモデルが適用されるのは、その動きがアフィン変換の形で定義できる剛体に限られているという点で、この方式はどちらかと言えば、制約的である。以下のセクションでは、アニメーション剛体に限定されない、より一般化された頂点位置の時間依存マトリックスを圧縮する方法について説明する。

【0036】

フリーフォーム変形 (Free-Form Deformations)

より一般化されたグラフィックスアーキテクチャによれば、剛体の動きの制約は、より一般化されたパラメータ化変形 (parameterized deformation) をサポートすることによって解消している。より一般化された3D動きモデルの例としては、フリーフォーム変形格子 (free-form deformation lattice) をもつ逆キネマチックリンケージ (inverse-kinematic linkage) がある。この形態の動きモデルは文字アニメーションで広く使用されている。次の式は、一般的時間依存ジオメトリマトリックスがこのより一般化された動きモデル用にどのように因数分解されるかを示している。時間依存ジオメトリマトリックスの係数としては、変形関数Fのマトリックス、一連の剛体V、および剛体の各々ごとに対応する一連のジオメトリ補間関数Gがある。

【0037】

【数6】

$$V_{FFD} = \overbrace{\begin{bmatrix} F_0^0 & \cdots & F_0^{R-1} \\ \vdots & \ddots & \vdots \\ F_{m-1}^0 & \cdots & F_{m-1}^{R-1} \end{bmatrix}}^{\text{変形}} diag(\overbrace{\hat{V}^0 \cdots \hat{V}^{R-1}}^{\text{剛体}})$$

【００３８】

左側のマトリックスの変形関数は、剛体を構成する頂点の集合の動きを定義するパラメータが少数になっているのが代表的である。各変形関数のこれらの動きパラメータは時間を通るようにスプラインされることが時にはあるが、ユーザ入力から計算されて、アニメーションシーケンス内のフレームに対する剛体の位置が決定されることもある。上式において、変形関数Fと対応する剛体

【００３９】

【数７】

$$\hat{V}$$

【００４０】

との乗算は、変形関数Fを剛体の頂点に適用していることを示している。

【００４１】

【数８】

$$V_{FFD} = \overbrace{\begin{bmatrix} f_0^0 & \cdots & f_0^{k(R-1)} \\ \vdots & \ddots & \vdots \\ f_{m-1}^0 & \cdots & f_{m-1}^{k(R-1)} \end{bmatrix}}^{\text{変形頂点}} diag(\overbrace{\begin{matrix} v_0^0 & \cdots & v_0^{R-1} \\ \vdots & & \vdots \\ v_{k-1}^0 & \cdots & v_{k-1}^{R-1} \end{matrix}}^{\text{剛体のパラメータ化座標}})$$

【００４２】

キー形状 (Key Shapes)

もう１つの有用な圧縮手法は、マトリックスを基底関数と重みに分解することである。この手法は、主要コンポーネント分析またはKL変換 (Karhunen-Loeve) といった、いくつかの名前で通っている。

【0043】

【数9】

$$V_{KL} = \overbrace{\begin{bmatrix} w_0^0 & \cdots & w_0^{k-1} \\ \vdots & & \vdots \\ w_{sf-1}^0 & \cdots & w_{sf-1}^{k-1} \end{bmatrix}}^{\text{重み}} \overbrace{\begin{bmatrix} \hat{v}_0^0 & \cdots & \hat{v}_0^{N-1} \\ \vdots & & \vdots \\ \hat{v}_{k-1}^0 & \cdots & \hat{v}_{k-1}^{N-1} \end{bmatrix}}^{\text{メッシュ基底ベクトル}}$$

【0044】

この変換によると、データの相関性が最大限になくなり、正規直交の基底（orthonormal basis）が得られる。しかし、KL手法はデータの非線形変換をキャプチャすることができない。z軸を中心にスピンしながらx方向に並行移動する剛体形状を想像してみると、この単純な回転と並行移動をキャプチャするように結合できる基底形状（basis shape）の集合が存在しない。

【0045】

スキンニング（skinning）

もう1つ広く使用されている手法は各頂点で「スキンニング」重みβjを使用して、移動する「ボーン(bone)」座標フレームC_iがメッシュをどのように変形するかを決定することである。代表例として、重みのほとんどはゼロであるので、頂点当たりの重みマトリックスはスパース（sparse）になっている。この手法の利点は、単一静止形状（rest shape）が、関心のある領域の近くに埋め込まれている座標フレームの集合によって変形できることである。初期フレームの逆がカレント座標フレームに事前付加されていると、「ボーン」座標フレームが得られる。

【0046】

【数10】

「ボーン」フレーム　頂点当たり重み　残余形状頂点

$$V_{Skin}=\overbrace{\begin{bmatrix} C_0^0 & \cdots & C_0^{R-1} \\ \vdots & \ddots & \vdots \\ C_{m-1}^0 & \cdots & C_{m-1}^{R-1} \end{bmatrix}}\overbrace{\begin{bmatrix} \beta_0^0 I & \cdots & \beta_0^{n-1} I \\ \vdots & \ddots & \vdots \\ \beta_{R-1}^0 I & \cdots & \beta_{R-1}^{n-1} I \end{bmatrix}}\overbrace{diag(v_0 \cdots v_{R-1})}$$

【００４７】

特殊ケース重み付け軌跡（Special-Case Weighted Trajectories）

時間依存ジオメトリは、3D空間内の選択したコントロールポイントの動きを通して、これらのコントロールポイントとメッシュ内の頂点とのアソシエーションと共に表現することもできる。参考文献としては、Brian Guenter、Cindy Marie Grimm、およびHenrique Sarmento Malvar（以下、Guenter他という）による係属中の米国特許出願第09/093,590号、発明の名称「3Dジオメトリ、表面表現のカラーと陰影および他のアニメーションオブジェクトをキャプチャし表現する方法およびシステム（Method and System For Capturing And Representing 3d Geometry, Color And Shading Of Facial Expressions And Other Animated Objects）」があるが、その内容全体は引用により本明細書に含まれている。Guenter他では、コントロールポイントの動きが関連頂点に適用されてメッシュを変形している。あるフレームから別のフレームへのコントロールポイントの動きは、軌跡（trajectory）と呼ばれている。

【００４８】

コントロールポイントをこのように使用することは、一般的に、種々のグラフィックスアプリケーションに応用可能であるが、Guenter他では、コントロールポイントは人間の顔を表している時間変化ジオメトリを符号化するために使用されている。Guenter他では、俳優の顔の動きは、基準（fiducial）「ドット」を付加し、その後ビジョン手法を使用してドットの動きを回復することによって回復されている。また、Guenter他では、静止位置で俳優の顔を表しているスタチック3Dメッシュを、従来の3Dキャプチャシステムを使用してキャプチャしている。Guenter他では、これらのドットをコントロールポイントとして使用し、次のようにドットの動きを適用してメッシュを変形している。

【0049】

【数11】

$$V_{Face} = V_0 + \overbrace{\begin{bmatrix} d_0^0 & \cdots & d_0^{k-1} \\ \vdots & & \vdots \\ d_{M-1}^0 & \cdots & d_{M-1}^{k-1} \end{bmatrix}}^{\text{ドット軌跡}} \overbrace{\begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^{N-1} \\ \vdots & & \vdots \\ \alpha_{k-1}^0 & \cdots & \alpha_{k-1}^{N-1} \end{bmatrix}}^{\text{ドット影響}}$$

【0050】

Guenter他では、この因数分解は、ドット軌跡の主要コンポーネントを計算し、その結果の係数をコード化することによってさらに圧縮されている。

【0051】

【数12】

$$V_{Face} = V_0 + \overbrace{\begin{bmatrix} w_0^0 & \cdots & w_0^{L-1} \\ \vdots & & \vdots \\ w_{M-1}^0 & \cdots & w_{M-1}^{L-1} \end{bmatrix}}^{\text{ドット係数}} \overbrace{\begin{bmatrix} b_0^0 & \cdots & b_0^{k-1} \\ \vdots & & \vdots \\ b_{L-1}^0 & \cdots & b_{L-1}^{k-1} \end{bmatrix}}^{\text{ドット基底}} \overbrace{\begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^{N-1} \\ \vdots & & \vdots \\ \alpha_{k-1}^0 & \cdots & \alpha_{k-1}^{N-1} \end{bmatrix}}^{\text{ドット影響}}$$

【0052】

この符号化方式では、ドット軌跡にキー形状を使用してから、ドットの動きを頂点の残余に適用している。

【0053】

セグメンテーション（segmentation）

最初のステップは、頂点マトリックスVのどの列をローカル座標系で符号化するかを決定することである。ある種のアニメーションでは、モデリングパッケージからこの情報が得られるようになっている。いくつかの現行アニメーションパッケージは「ボーン」と「スキンニング」を使用し、座標フレームは最終頂点位置が得られるように重み付けされている。汎用アニメーション（シミュレーション、シェープカメラ（shape camera）などの出力）では、分解を良好にするために入力頂点マトリックスが分析されている。「ボーン」の数をランタイムシステ

ム用に変更する必要があるときは、自動分析によると便利である。このクラスタリング問題は、学習、ベクトル量子化、および他の圧縮手法でよく起こっている。本件のケースでは、ベクトルは頂点マトリックスの列から構成されている。問題は、クラスタがいくつ必要であり、どのクラスタを各クラスタに所属させるかを決定することである。もう1つの問題は、変形のクラスをクラスタごとに決定することである。

【0054】

クラスタセグメンテーション問題は次の通りである。頂点マトリックスVが与えられているとき、クラスタの数nclusterと、各頂点のクラスタ割り当てのリストが戻される。複数の割り当てが許されるので、各頂点はあるクラスタの集合に所属し、関連の重みをもつことができる。各クラスタは関連の時間変化座標フレームCをもち、これは所与であるか、あるいはクラスタのメンバである頂点に基づいて計算されたものである。

【0055】

【数13】

$$v^j(t) = \sum_k w_k^j C_k(t)$$

【0056】

プロトタイプのセグメンテーションアルゴリズムでは、オリジナルメッシュのトライアングルをベースにした貪欲クラスタリング手法（greedy clustering approach）を使用している。シードトライアングル（seed triangle）の集合はランダムに選択されている。トライアングルは、必要ならば、すべてをシードトライアングルとして使用できるが、以下に説明する実験では、メッシュ内のオリジナルトライアングルの約10%が使用されている。シードトライアングルの座標フレーム軌跡が比較され、所与の許容範囲内にあれば、クラスタは結合される。頂点の軌跡は、その結果得られたクラスタの各々のローカル座標系に投影され、軌跡を通るマッチング（match）の品質別に分類される。

【0057】

ジオメトリック変換コード化 (Geometric Transform Coding)

ジオメトリック変換コード化 とは、3Dジオメトリの動きをジオメトリック変換で近似化し、その後、アニメーションシーメンス全体にわたる選択されたタイムサンプル期間の、変換された3Dジオメトリと3Dジオメトリの実際位置との差分を符号化することによって時間依存ジオメトリを圧縮する方法のことである。これらのタイムサンプルはアニメーションシーケンスの時間に対応し、そこでは、オブジェクトの位置が更新されている（フレームごとに更新されるとは限らない）。アニメーションシーケンスの時間依存ジオメトリは一連の3Dメッシュとして表され、各々はアニメーション内の特定タイムサンプルでの頂点位置の集合を示している。この形態の時間依存ジオメトリの具体例として、上述した頂点位置マトリックスVがある。このジオメトリック変換コード化方法は基底メッシュを選択することから始まる。次に、基底メッシュとアニメーション内のメッシュの各々との間の、ジオメトリック変換が決定される。時間依存ジオメトリを圧縮するために、この方法によれば、変換された基底メッシュと実際のメッシュとの間の差分が決定され、この差分は残余（residual）と呼ばれる。

【0058】

ジオメトリック変換コード化手法は、時間依存ジオメトリがアニメーション剛体から構成されている特殊ケースと競合関係にある。この特殊ケースでは、基底メッシュは剛体に対応し、ジオメトリック変換は、アフィン変換や格子フリーフォーム変換のように、剛体に適用される動きモデルに対応している。この概念を示すために、一緒にアニメーション化される頂点のブロック、すなわち、アフィン変換やフリーフォーム変形格子をもつ逆キネマチックリンケージのように、単純化クラスの移動ジオメトリのどちらかにおける剛体に対応している列のブロックを考えてみることにする。このケースでは、変換されたベース剛体とカレントメッシュの間にどれだけのひずみがあるかは、残余から分かるようになっている。以下の式は、変形関数Fが剛体

【0059】

【数14】

$\hat{V}$

【0060】

に適用された場合の、残余の計算を示している。

【0061】

【数15】

$$\Delta V^j = V^j - \begin{bmatrix} F_0^j \\ F_1^j \\ \vdots \\ F_{m-1}^j \end{bmatrix} \left[\hat{V}_j\right]$$

【0062】

アニメーションシーケンスの時間依存ジオメトリが変形関数を剛体に適用することによって作られる場合は、ひずみがなくなる。以下に説明するジオメトリック変換コード化方法がもっと一般的であるのは、動きをもっと一般的な形で表現でき、変換関数または時間依存ジオメトリの剛体部分がコード化時に分かっていないような、任意のメッシュを符号化するのに適応できるからである。

【0063】

時間依存ジオメトリを表している所与のデータ集合を圧縮するには、ジオメトリック変換方法では、変形パラメータ、ベースメッシュまたはメッシュ、および残余を量子化し、符号化している。時間依存ジオメトリが単純化サブクラス（剛体または格子FFD）の1つに合致するときは、残余はゼロであるので、符号化するときのオーバヘッドは非常に低くなる。

【0064】

ジオメトリックコード化方法は単独で使用することも、時間依存ジオメトリマトリックスVのコヒーレンスを活用するために下述する他の圧縮手法と併用することも可能である。時間依存ジオメトリマトリックスの時間的および空間的コヒーレンスをさらに活用するための1つの方法は、マトリックスVの行と列の中からコヒーレンスを特定し、符号化することである。時間依存ジオメトリをコード化する方法は、以下で説明するように、単純化された動きモジュールに内在する

圧縮を改善し、しかも、現行3Dグラフィックスレンダリングアーキテクチャとの互換性を維持したまま、もっと複雑な動きモデルに応用可能である。

【００６５】

図１は、メッシュ変換コーダの例を示すブロック図である。図１には、時間依存ジオメトリがどのように符号化されて伝送またはストアされた後、どのように復号化されるかを示すために、圧縮器（compressor）２０と伸張器（decompressor）４０の両方が示されている。圧縮器２０への入力には、３種類ある。時間の経過と共に変化するメッシュ頂点の位置を表しているマトリックスVと、ベースメッシュVと、ベースメッシュを、時間依存マトリックスV内のメッシュとマッチングするジオメトリック変換（" xfm" ）である。圧縮器はジオメトリック変換の変換パラメータ、ベースメッシュ、および残余を量子化し、符号化する。伸張器で再構築されるメッシュのひずみを最小限にするために、圧縮器は量子化/量子化解除変換（quantized/de-quantized transformation）とベースメッシュパラメータを使用して残余を計算する。

【００６６】

このエンコーダでのベースメッシュは、剛体を表す頂点のメッシュになることを目的としている。しかし、一部の実施形態では、ベースメッシュはコーダに与えられないので、これは時間依存ジオメトリVのストリームから導き出す必要がある。このような場合には、ベースメッシュは、マトリックスVの行をベースメッシュとして使用するか、Vの複数行の平均をベースメッシュとしてとるか、あるいは変換パラメータと残余のための結合帯域幅が最小になるようにベースメッシュを事前に計算することで、計算することができる。後者の方法によると、変形パラメータとベースメッシュの両方は同時に非線形的に最適化される。

【００６７】

図１に圧縮器への入力として示されているジオメトリック変換（" Xfm" ）は、選択したタイムサンプルインクリメントでベースメッシュの位置を新しい位置に変換するために使用されるジオメトリック変換パラメータの集合である。従来のグラフィックスレンダリングパイプラインでは、選択したタイムインクリメントはアニメーションシーケンス内のフレームに対応しているのが代表的である。

しかし、もっと一般的には、タイムインクリメントは、変化するレートでオブジェクトの更新を可能にするレンダリングアーキテクチャと互換性をもつように、タイムサンプルに対応させることも可能である。変換パラメータは、タイムサンプルごとにベースメッシュを新しい位置に変換するために使用されるので、各サンプルは対応する変換パラメータの集合をもっている。ベースメッシュが剛体を表している場合には、変換パラメータは、アフィン変換や一般的フリーフォーム変換のように、剛体をそのローカル座標系からアニメーショングラフィックスシーンのワールド座標に変換するジオメトリック変換を表している。

【0068】

圧縮器20は、変換パラメータとベースメッシュの3D位置値を、それぞれ量子化する量子化モジュール22、24を備えている。また、圧縮器は変換パラメータとベースメッシュの3D位置値を量子化解除するために使用される量子化解除器26、28も備えている。圧縮器20に示されている変換モジュール30は、量子化解除変換パラメータを量子化解除3D位置値に適用して、時間依存ジオメトリのカレント位置に近似化している変換メッシュを出力する。その後、減算器モジュール32は変換ベースメッシュの値と、マトリックスVの対応する要素との差分を計算する。具体的には、減算器モジュールは、量子化解除され、変換されたベースメッシュの頂点と、カレントタイムサンプルのときの頂点位置マトリックスVの行の頂点との差分を計算する。

【0069】

圧縮器20の出力には、タイムサンプルごとの量子化変換パラメータと量子化ベースメッシュが含まれており、これは圧縮データストリームの先頭で一度送信される。この出力には、量子化器34で計算された量子化残余も、タイムサンプルごとに含まれている。従って、圧縮データストリームは、アニメーションシーケンスに対してベースメッシュを一度符号化し、アニメーションシーケンス内の各フレームのように、アニメーションシーケンスに含まれる複数のタイムサンプルの残余で変換パラメータを符号化する。圧縮データストリームは、将来の使用に備えて永続メモリにストアしておくことも、トランスミッタに送信し、そこから別のコンピュータやレンダリングデバイスに送信することも可能である。

【００７０】

伸張器４０は、選択したタイムインクリメントのときの時間依存ジオメトリを、圧縮データストリームから再構築する。具体的には、伸張器はアニメーションシーケンス内のタイムサンプルに対して、頂点位置のマトリックスを１行ごとに再構築する。伸張器に含まれる量子化解除器４２、４４は、残余、変換パラメータ、およびベースメッシュの3D位置データをそれぞれ量子化解除する。ジオメトリック変換モジュール４８はカレントタイムサンプルの量子化解除変換パラメータを量子化解除ベースメッシュに適用して、変換ベースメッシュを計算する。加算器モジュール５０はカレントタイムサンプルの変換ベースメッシュを、そのタイムサンプルのときの、対応する量子化解除残余と結合し、時間依存ジオメトリのカレント3D位置を計算する。

【００７１】

伸張器４０はオンラインレンダリングアプリケーションで使用できる。その場合には、圧縮時間依存ジオメトリが取り出されたあと、頂点位置の集合は時間的余裕をもって再構築されてからジオメトリが3Dグラフィックスレンダリングサブシステムに渡され、そこでアニメーションシーケンスでのレンダリングが行われることになる。

【００７２】

変換マッチング (Transformation Matching)

ある種のアプリケーションでは、時間と共に変化する変換パラメータは、時間依存ジオメトリのコード化が開始される前に使用可能になっていない。このことが特に該当するアプリケーションとしては、アフィン変換やフリーフォーム変形などの、標準的ジオメトリック変換を使用して変換された単純な剛体になっていない時間依存頂点位置の集合を表している、より一般化されたマトリックスのコード化を行うアプリケーションがある。このようなアプリケーションでは、圧縮器はベースメッシュを、アニメーションシーケンス内の選択したタイムインクリメントのときのカレントメッシュに最良に合致させる一連の変換を得ている。以下では、一般的で、任意の動きをもつ時間依存ジオメトリを表しているマトリックスの変換パラメータを導き出すコーダについて説明する。以下に説明する図示

の例では、アフィン変換を導き出すことが示されているが、これと同じ手法は他の動きモジュールにも応用可能である。

【0073】

図2は、変換マッチングを行って時間依存ジオメトリから変換パラメータを導き出すベースメッシュ変換コーダを示すブロック図である。図1と同じように、図2には、圧縮器モジュール60と伸張器モジュール62が示されている。図を単純化するために、図1に示す量子化器モジュールと量子化解除器モジュールは、図2には破線62－68で示されている。

【0074】

圧縮器60への入力としては、ベースメッシュV_0と、時間依存ジオメトリを表している頂点位置のマトリックスVとがある。ベースメッシュが剛体を表している特殊ケースとは対照的に、より一般的なケースでのベースメッシュはコード化を始める前に分かっている必要がない。ベースメッシュはマトリックスVから導き出すことができる。例えば、ベースメッシュはVの行を選択するか、Vの行の平均をとるか、あるいは上述したように、変換パラメータと残余の結合帯域幅が最小になるようにベースメッシュを再計算することによって選択することができる。

【0075】

ベースメッシュと同じように、変換パラメータも頂点位置のマトリックスから導き出すことができる。図2に示すように、圧縮器60に置かれた変換マッチングモジュール（transform match module）70は、ベースメッシュを、マトリックスV内の時間依存メッシュの各々とに最良に合致させるジオメトリック変換を決定する。マトリックスVの行の各々は、選択した時間におけるメッシュに対応している。変換マッチングモジュール70は、マトリックスVの行にストアされたメッシュの位置に近似化するようにベースメッシュを変換するために使用されるジオメトリック変換の変換パラメータを計算する。

【0076】

圧縮器60は変換モジュール72を含み、このモジュールは量子化解除変換パラメータを量子化解除メッシュパラメータに適用し、カレントメッシュの位置に

近似化するようにする。その後、減算器モジュール７４は、変換ベースメッシュとカレントメッシュとの間の3D位置の差分を計算し、残余を出力する。残余、変換パラメータ、およびベースメッシュパラメータは、破線６４、６６、６８で示すように量子化され、送信される。ベースメッシュは、代表的なアニメーションシーケンスでは一度送信されるだけである。変換パラメータと残余は符号化されるアニメーションのタイムサンプルごとに計算される。代表例では、このタイムサンプルはアニメーションシーケンスのフレームに対応している。

【００７７】

図１と同じように、図２の伸張器６２は圧縮データストリームを復号化し、時間依存ジオメトリのストリームを再構築するために使用されるコンポーネントを表している。破線６４－６８で示す固有の量子化解除モジュールのほかに、伸張器６２は変換モジュール８０と加算器モジュール８２も備えている。変換モジュールは量子化解除変換パラメータを量子化解除ベースメッシュに適用し、選択したタイムサンプルに対する変換ベースメッシュを計算する。加算器モジュール８２は変換ベースメッシュを受け取り、その時間の間にそれを残余メッシュと結合してメッシュを再構築する。変換モジュール８０と加算器モジュール８２は、他の変換パラメータと関連残余メッシュの集合ごとに上記プロセスを繰り返す。

【００７８】

変換マッチングモジュール７０がどのように実現されるかは、その一部として、ベースメッシュとマトリックスVの対応するメッシュとの間の変更位置に近似化するために使用される動きモデルによって決まる。例えば、3Dの動きがアフィン変換モデルに基づいて推定される場合は、変換マッチングモジュール７０は、選択した時点において変換ベースメッシュと対応するメッシュとが最も緊密に合致しているものが得られるようにアフィン変換係数を計算する。変換係数を計算する式は次のように表すことができる。

【００７９】

【数１６】

$$A_k \hat{V}_0 = V_k$$

【0080】

上式において、A_k は望ましい4 x 4マトリックスであり、これは、ベースメッシュV_0 を、一般的マトリックスVのカレント行であるV_k に変換するために使用される。最良の最小2乗解 (best least-squares solution) を表している変換係数を求める1つの手法は、ベースメッシュV_0 の特異値分解 (singular value decomposition) を計算し、ベースメッシュの分解をVのカレント行に適用することである。この手法では、マトリックスが大きいと計算コストが高くなる。この計算の複雑さを軽減するために、変換マッチングモジュール70は、以下で説明する最適化の1つまたは2つ以上を使用するように適応させることができる。1つの最適化は、ベースメッシュとカレントメッシュに低詳細レベルのマトリックスを使用して計算を行うことである。低詳細レベルのマトリックスを使用すると、最良の最小2乗解の近似値を計算し、それを完全マトリックスに適用することができる。

【0081】

もう1つの最適化は正規方程式 (normal expression) を使用することであるが、そこでは、ベースメッシュの4 x 4適合マトリックス、および n x 4マトリックスと適合マトリックスとのマトリックス積を累算し、反転することが行われている。この手法を数学的に示すと、次の2式のようになる。

【0082】

【数17】

$$K = V_0^T \left(V_0 V_0^T\right)^{-1}$$
$$A = VK$$

【0083】

正規方程式の手法は完全方程式系の解を求めて、変換係数を決定するほどには堅固でないが、実用的にはその働きは十分である。この手法が完全系で解を求めるほど正確でないのは、小さな部分空間に投影するとき情報が失われるからである。しかし、変換係数はメッシュの動きの近似にすぎないため、正規方程式の手法でも十分に正確である。セグメント化頂点ブロックのフレームごとに、最良合

致 (matching) アフィン変換を計算するために、プロトタイプでは次の方法が採用されている。

【0084】

1) 静止形状V_o (これは頂点マトリックスの最初の行またはモデルの代表的ポーズのどちらかである) を使用して、合致マトリックス $K = V_o^T (V_o V_o^T)^{-1}$ を計算する。

【0085】

2) アニメーションのフレームごとに、最良合致アフィン変換を$Aj = V_j K$として計算する。

【0086】

3) 頂点グループが縮退 (degenerate) (つまり、すべての点が平面内にある) であるときは、$V_o V_o^T$ マトリックスは特異であるか、状態が乏しくなっている。このケースでは、静止形状頂点V_oとフレーム当たり頂点V_jを増加するために追加頂点が追加され、縮退を取り除いている。この追加頂点は、各トライアングルの法線に沿って各トライアングルの中心をオフセットすることによって、頂点に関連するトライアングルから導き出されている。

【0087】

図2に示す手法で使用されているジオメトリック変換がアフィン変換であれば、カレントフレームのメッシュの位置に近似化するために使用されるジオメトリック変換を、そのフレームのビューイング変換と一緒に結合することが可能である。ジオメトリック変換コーダは、図3に示すように改良すれば、ジオメトリック変換をビューイング変換と結合することが可能になる。

【0088】

図3は、ジオメトリック変換ではなく逆ジオメトリック変換 (inverse geometric transform) を符号化する代替ベースメッシュ変換コーダを示すブロック図である。このタイプのコーダでは、各タイムサンプルの残余は、カレントメッシュのローカル座標で変換ベースメッシュをカレントメッシュから減算するのではなく、変換ベースメッシュの座標で計算される。この計算式は次に示すとおりである。

【００８９】

【数１８】

$$\Delta V_k^j = \left(a_k^j\right)^{-1} V_k^j - \hat{V}_0^j$$

【００９０】

上式において、残余パラメータΔV_kは、1) ベースメッシュの座標空間に変換された頂点位置のマトリックスのカレント行と、2) ベースメッシュV_0との差分として計算される。変換ベースメッシュをカレントメッシュにマッチングするのではなく、変換パラメータはカレントメッシュをベースメッシュにマッチングする。

【００９１】

図３に示す逆変換コーダに含まれる圧縮器１００と伸張器１０２は同じ入力データに作用し、図２のメッシュ変換コーダと同じようなコンポーネントをもっている。具体的には、圧縮器の変換マッチングモジュール１０４は、変換されたカレントメッシュをベースメッシュと合致させる変換パラメータを計算する。その後、逆変換モジュール１０６は量子化解除変換パラメータをカレントメッシュに適用し、カレントメッシュをベースメッシュ座標に適用して残余を計算し、減算器モジュール１０８は、変換されたカレントメッシュとベースメッシュの対応する要素間の差分を計算する。圧縮器は入力マトリックスVの行ごとに上記プロセスを繰り返し、圧縮ジオメトリストリームを出力する。

【００９２】

伸張器は量子化残余、量子化変換パラメータ、および量子化ベースメッシュに作用する。時間依存ジオメトリ内の各メッシュを再構築するために、伸張器１０２に含まれる加算器モジュール１１０はベースメッシュの量子化解除パラメータを量子化残余と結合する。伸張器に含まれるジオメトリック変換モジュール１１２は、カレントフレームのビューイング変換を、圧縮器で計算された3D変形変換と結合する量子化解除変形を入力として受け取る。変換モジュール１１２はカレントメッシュを、ベースメッシュ座標からワールド座標に変換し、その後ワールド座標からビューイング座標に変換する。この方法によると、圧縮器の変換モジ

ユールは変換マッチングで計算された3D変換を、グラフィックスレンダリングパイプラインの標準モデリングおよびカメラ変換と結合することができる。

【0093】

メッシュフィードバック (Mesh Feedback)

これまでのセクションで説明したように、ある種の時間依存ジオメトリのベースメッシュは剛体を表す事前定義のメッシュであり、他方、他のアプリケーションでは、メッシュは一般的なマトリックス位置の頂点から導き出されている。時間依存ジオメトリの経時的コヒーレンスを利用するために、ジオメトリック変換コーダは、先行フレームのメッシュをカレントフレームのベースメッシュとして使用するように適応化することができる。この方法によると、先行フレームのメッシュを使用するフィードバックループは、メッシュがカレントフレームのどこに置かれるかを予測する手段（predictor）として使用されている。ここで注意すべきことは、この種の予測が、頂点位置のマトリックスで行われる行予測と異なっているのは、変換がN次元ベクトル空間ではなく、3D座標空間で行われるためである、ということである。しかし、以下で詳しく説明するように、行予測と列予測を使用すると、どちらの場合も、残余やベースメッシュのような、3D位置データのマトリックスをさらに圧縮することができる。

【0094】

図4は、先行タイムサンプルで計算されたメッシュを、カレントタイムサンプルのベースメッシュとして使用するメッシュ変換コーダを示すブロック図である。図4には、この種のコーダ用として圧縮器120と伸張器122の両方が示されている。このコーダでは、各カレントフレームのベースメッシュは先行フレームからの構築メッシュである。

【0095】

他のメッシュコーダと同じように残余を計算することのほかに、圧縮器は先行タイムサンプルからのメッシュを再構築したあと、その再構築メッシュのコピーを保存している。第1フレームのメッシュの予測手段として初期メッシュから始めて、変換マッチングモジュール124は最良変換を見つけ、先行フレームの近似頂点V' を次フレームのメッシュにマッピングする。カレントメッシュは、ア

ニメーションシーケンス全体のシングルベースメッシュよりも先行フレームのメッシュに類似しているのが普通であるので、この手法によって得られる残余値は小さくなる傾向がある。

【0096】

変換マッチングモジュール124は、「ディレイ」メモリに一時的にストアされた近似メッシュに作用し、先行メッシュとカレントメッシュが最良適合する変換パラメータを見つける。図2および図3と同じように、破線は量子化器と量子化解除器のペアを示している。圧縮器は変換マッチングモジュール124からの変換パラメータを量子化する。量子化パラメータは、圧縮データストリーム128の一部となり、圧縮器で残余を計算するために使用される前に量子化解除器にも入力される。

【0097】

変換モジュール130は量子化解除変換パラメータを先行フレームの近似メッシュに適用し、カレントフレームのメッシュの位置を推定する変換メッシュを計算する。圧縮器の減算器モジュール132は残余を、先行フレームの変換メッシュとカレントメッシュとの差分として計算する。この残余は量子化され、量子化残余は、量子化変換パラメータ128と共に圧縮データストリームの別部分となる。

【0098】

圧縮器のフィードバックループはカレントメッシュの近似メッシュを再構築し、カレントフレームの近似メッシュをメモリ126に一時的にストアしておく。具体的には、加算器モジュール136はカレントフレームで計算された残留3D位置データを、変換バージョンの先行フレームと結合し、その結果のカレントメッシュの近似メッシュをメモリ126にストアしておく。圧縮器は、あるタイムインクリメント、例えば、フレームのときの、頂点のカレント近似頂点のコピーを保存しておき、時間依存ジオメトリVで次タイムインクリメントのときの、メッシュを予測する手段として使用できるようにする。

【0099】

伸張器122にも、最近に計算された頂点をあるタイムインクリメントだけ遅

延させるフィードバックループがあるので、次の頂点集合を予測する手段として使用できるようになっている。伸張器の変換モジュール１４０は量子化解除変換パラメータを、以前に構築され、メモリ１４２に一時的にストアされていたメッシュに適用する。変換モジュール１４０の出力はカレントメッシュに変換された先行フレームのメッシュである。加算器モジュール１４４はこの変換メッシュを、圧縮データストリームからの、カレントフレームの量子化解除残余と結合し、カレント近似頂点V' を構築する。

【０１００】

図３を参照して上述した逆変換コード化方法は、図４に示すフィードバックコーダで使用することも可能である。このケースでは、変換マッチングモジュール１２４はカレントメッシュ入力を、先行フレームの近似メッシュの座標に変換する。その後、圧縮器は先行フレームの座標に変換されたカレントメッシュと先行フレームの近似メッシュとの差分として、残余を計算する。圧縮器と伸張器は、先行フレームの近似メッシュを変換カレントメッシュと結合することによって、残余からカレントメッシュを再構築する。図３の逆メッシュコーダと同様に、その結果の3D位置データは、変換マッチングモジュール１２４で計算された変換パラメータを使用して、カレントフレームでその位置に変換される。しかし、図３のコーダとは異なり、3D変換はカメラ変換と結合することはできない。カレントメッシュは、カメラ変換なしで変換し、次フレームのメッシュの位置を予測する際に使用されるように別にストアしておかなければならない。

【０１０１】

列と行の予測とソート

時間依存ジオメトリのマトリックスでジオメトリックと時間的コヒーレンスを利用するもう１つの方法は、マトリックスの行と列で予測を行うことである。上述した圧縮方法と同じように、時間依存ジオメトリのストリームは3Dジオメトリック位置データのマトリックスで表すことでき、そこでは、各行はアニメーションシーケンスのフレームのような、タイムサンプルに対応し、各列は3D位置と関連付けられている。オリジナルマトリックスを圧縮マトリックスから減算すると（つまり、上述した方法を使用すると）得られる残余マトリックスはオリジナル

マトリックスと同じ形態になっているので、このセクションで説明している圧縮手法は、オリジナル頂点マトリックスに関する残余マトリックスにも十分に適用される。時間的予測は、各行を別の行または参照行（reference row）とペアにし、2行の各対応する要素の間の差分を計算し、例えば、差分値を量子化することによって2行の間の差分を符号化することによって行うことができる。ジオメトリック予測は、類似の手法を用いて列についても行うことができる。また、各列に関連する3D位置間のジオメトリック接続性は独立に指定されるのが普通であるので、列は、隣接列の対応する要素の値が可能な限り類似するようにソートすることができる。

【0102】

図5は、圧縮器160と伸張器162の両方を備えたマトリックス予測コーダの例を示すブロック図である。圧縮器160は時間依存ジオメトリストリームに作用するが、これは、このケースでは、頂点位置のマトリックスVになっている。圧縮器に含まれるソートモジュール162は、隣接列/行の対応する要素が、オリジナルマトリックスよりも類似するように列または行（または両方）を位置付ける。具体的には、ソートモジュール162は、隣接頂点パス（経路）が可能な限り類似するように列をソートする。このジオメトリでのソーティングプロセスは、コヒーレントな動きをもつジオメトリの部分を一個所に置くようにする。3Dジオメトリが一連の剛体からなるようなケースでは、列は一緒に移動する剛点グループになるように配列される。

【0103】

もっと一般化された頂点マトリックスでは、ソートモジュール162は、列の類似性程度に作用して類似の動きをもつ列グループを見つける。この類似性程度の1つとして、平均除去列（mean-removed columns）の内積（inner product）が使用できる。

【0104】

類似性程度を計算するには、

1）マトリックス内の各列の平均をとり、その結果を行ベクトル

【0105】

【数19】

$$\overline{V}$$

【0106】

にストアし、

2）列平均値を各行

【0107】

【数20】

$$\hat{V}_i = V_i - \overline{V}$$

【0108】

から減算する。

【0109】

その後、ソータはその結果の類似性程度を使用して列をソートすることができる。

【0110】

プロトタイプでは、列iとjの類似性は2つの方法で計算されている。生（raw)の列ドット積（raw column dot product）$V^i \cdot V^j$ 、および列要素間の二乗距離（squared distances）の和（$\Sigma (V_k^i - V_k^j)^2)^{1/2}$ である。最も類似している頂点の列は相互に隣接して置かれる。左から始まって右に移動しながら、列ソータは残余の列から最も類似する列を見つけていき、カレント列のちょうど右になるようにその列をスワップする。トライアングルリストは頂点インデックスを使用し、各頂点は列で表されているので、マトリックス内のロケーションと出力時の列の使用との間にはインディレクション(indirection) のレベルがある。このようにすると、デコーダに気づかないで列の再配列が行われることになる。エンコーダは、頂点マトリックス内のソートされた列に対応するように、トラインアングルリスト内のインデックスに番号を付け直す。

【0111】

行のソーティングも、列のソーティングと同じように行うことができる。

【0112】

類似の経路を移動する頂点は類似するものとみなされる。この特定ソートでは、2D表面で線形的ソート（linear sort）が行われるので、得られる最良結果は、Peano曲線に類似する表面を満たす頂点マトリックスを通る経路になっている。なお、Peano曲線は「空間を満たす」曲線の一種である。言い換えれば、頂点は2Dメッシュ上にあるので、利用できる隣接間のコヒーレンスは単純な1Dソートによる場合よりも向上するが、1Dソートでも、コヒーレンスが向上するので、伸張器は低コストになる。エッジ縮小（edge contraction）で隣接頂点をクラスタ化することによって列間のコヒーレンスを向上するソート手法は以後のセクションで詳しく説明する。

【0113】

ソートモジュール１６３は、マトリックスの行/列をソートして、隣接行/列のコヒーレンスを向上するプロセスを表しているのが一般である。ソートモジュール１６３は列データのソートを行うと、マトリックスのジオメトリック接続性データを更新し、ソートされたマトリックス内の該当列を正しく指すようにする。ジオメトリック接続性データ１６４は、マトリックス内の頂点位置間の接続性を定義している。データは、１つに接続されたメッシュ内の頂点の部分集合を定義しており、その構造内の頂点位置とマトリックスV内の対応する頂点位置の間のレファレンス（参照）も維持している。ソートモジュール１６３は列の順序を変更するとき、接続性データ内の頂点位置がマトリックスV内の該当列を指すように接続情報を更新する。

【0114】

空間的予測（spatial prediction）のほかに、圧縮器１６０は頂点位置のマトリックスの行について時間的予測（temporal prediction）も行う。大部分のケースでは、マトリックスV内の隣接行は、隣接行が最も類似している、対応する要素を含むようにすでに配列されている。しかし、隣接行がオリジナル入力マトリックスよりも類似するように行をソートすることも可能である。また、参照行、つまり、他の行の動きを予測する基礎として使用される参照行を選択することも可能である。行がソートされるとき、ソートモジュール１６３は行間の時間的

関係を維持しているデータ構造１６５を更新する。このデータ構造を使用すると、メッシュデータを正しい時間的順序で再構築できるかを確かめることができる。

【０１１５】

圧縮器１６０は、参照行/列と別の列/行間の各対応する要素の間の差分を計算することによって行/列予測を行う。ソートが完了すると、予測器モジュール１６６は、行予測を行うときは、ペアの隣接行内の各対応する要素の間の差分を計算し、列予測を行うときは、隣接列内の各対応する要素の間の差分を計算する。予測器モジュール１６６の出力は差分値のマトリックスである。これらの差分値は量子化モジュール１６８で量子化することができる。

【０１１６】

圧縮データストリームを伸張（decompress）するために、図５の伸張器１６２は圧縮器１６０のオペレーションと逆のことを実行する。具体的には、量子化解除器１７０は差分値のマトリックスを量子化解除する。その後、差分値のマトリックスは再構築され、ソートされたマトリックスのオリジナル行と列が逆予測器モジュール１７２で計算される。再配列モジュール（reorder module）１７４は列予測のためにソートされた場合は列を再配列し、行予測のためにソートされた場合は行を再配列する。ジオメトリック接続性データ１７６は列を再配列するために使用される。同様に、時間的データ１７８で表されている行の配列は、ジオメトリックデータが正しい時間シーケンスにあるかを確かめるために使用される。

【０１１７】

図５の上例では、頂点位置のマトリックスVが対象になっているが、行と列の予測は他の形態の時間依存ジオメトリマトリックスで行うことも可能である。例えば、以下で詳しく説明するように、時間依存ジオメトリックデータのマトリックスは、主要コンポーネント分析を使用して、基底ベクトルのマトリックスと別の重みマトリックスに分解することができる。このケースでは、行/列予測は、重みを表すマトリックスで行うことができる。メッシュコーダで計算された残余値のマトリックスとベースマトリックスで予測を使用することも可能である。

【0118】

基底分解コーダ (Basis Decomposition Coder)

時間依存ジオメトリを表すマトリックスを圧縮するもう１つの方法は、主要コンポーネント分析を使用してマトリックスを基底関数 (basis function) と重みに分解することである。マトリックスの最良基礎ベクトル集合を見つける手法は、例えば、PCA (principal components analysis—主要コンポーネント分析)、KL変換 (Karhunen-Loeve)、SVD (singular value decomposition—特異値分解) などのように、いろいろな名前で通っている。SVDは頂点マトリックスVをUDWに因数分解している。ここで、UとWは正規直交マトリックス (orthonormal matrix) であり、$D = \mathrm{diag}(s_0, s_1, s_2, \ldots)$ はサイズ別にソートされた特異値 (singular value) の対角マトリックス (diagonal matrix) である。特異値のサイズは対応する基底ベクトルの大きさを示している。基底ベクトルは特異値s_iと行W_iによって与えられる。各列U^iはフレーム当たりの対応する重みを示している。次式は、頂点位置のマトリックスがどのようにして基底ベクトルと重みのメッシュに分解されるかを示している。

【0119】

【数21】

$$V_{M\times N} = \overbrace{\begin{bmatrix} w_0^0 & \cdots & w_0^{k-1} \\ \vdots & & \vdots \\ w_{M-1}^0 & \cdots & w_{M-1}^{k-1} \end{bmatrix}}^{\text{重み}} \overbrace{\begin{bmatrix} \hat{v}_0^0 & \cdots & \hat{v}_0^{N-1} \\ \vdots & & \vdots \\ \hat{v}_{k-1}^0 & \cdots & \hat{v}_{k-1}^{N-1} \end{bmatrix}}^{\text{メッシュ基底ベクトル}}$$

【0120】

左側のマトリックスは重みを表し、重みは係数とも呼ばれる。頂点位置のオリジナルマトリックスと同じように、上に示す重みのマトリックスは、タイムインクリメントに対応する行とジオメトリック位置を表す列をもっている。同様に、メッシュ基底ベクトルを表している右側のマトリックスは、各タイムインクリメントの行と基底ベクトルを表す列を含んでいる。重要な基底ベクトルの数が少なければ、最も重要な基底ベクトルだけを符号化し、送信し、そのあとフレーム当

たりの重みと、明示的に符号化されなかった基底ベクトルの残りからの残余とを加えたものを送信するようにすると、良好な圧縮が得られる。

【0121】

主要コンポーネント分析は、3Dオブジェクトではフルサイズの頂点位置マトリックスで行うことができるが、特に複雑なオブジェクトでは、フルサイズマトリックスを分解すると、計算コストが高くなる。従って、フルサイズの特異値分解 V = UDW (Uはサイズnframe x nframe、Wはサイズnvertex x nvertex) を計算する方法もあるが、もっと良い方法は、マトリックスの処理前にメッシュ単純化を行ってから、基底分解を行うことである。このケースでは、オリジナル頂点マトリックスVは空間的（プログレッシブメッシュ法の使用による）と時間的（結び目削除（knot deletion）またはウェーブレット（wavelet）符号化の使用による）にスムーズなバージョンV_sが得られるまでフィルタに通され、得られたバージョンは$V_s = U_s D_s W_s$として因数分解されている。$D_s W_s$で与えられた基底ベクトルは、次に、プログレッシブメッシュ頂点分割レコードによって拡張されるが、これはファイン詳細（fine-detail）メッシュの基礎を得るために必要になるものである。同様に、U_sによって与えられる重みは、詳細レコード（結び目挿入またはウェーブレット詳細によって与えられる）によって時間的に拡張される。

【0122】

もう1つの方法は、特異値分解に対してスモールランク近似（small rank approximation）、$V_k = U_k D_k W_k^T$ (U_kはサイズnframe x K、V_kはサイズK x nvertex）を反復的に計算できる近似SVDコーダを使用することである。

【0123】

図6は、主要コンポーネント分析を使用して時間依存ジオメトリストリームを圧縮するメッシュ基底コーダを示すブロック図である。図6に示すメッシュ基底コーダは、頂点位置のマトリックスVをメッシュ基底ベクトルの集合に投影し、その後基底係数と残余を圧縮する。圧縮器200では、基底投影モジュール202はカレントメッシュを基底係数と基底ベクトルに分解する。量子化モジュール204は基底係数を量子化し、これは圧縮データストリーム206の一部になる

。量子化解除モジュール２０６はカレントメッシュの基底係数を量子化解除する。基底合成モジュール（basis synthesis module）２０８は、量子化解除基底係数と基底ベクトルからカレントメッシュを再構築する。圧縮器で残余を計算するために、減算器２１０は、再構築メッシュとカレントメッシュの間の、対応する各要素の差分を計算する。最後に、量子化器２１２は残余を量子化し、量子化された残余は量子化された基底係数と共に圧縮データストリーム２１４の第２部分となる。

【０１２４】

圧縮データストリームを伸張するために、伸張器２２０は、コード化残余と基底係数からカレントメッシュを再構築する。量子化解除器２２２は基底係数を再構築し、量子化解除器２２４は残余を再構築する。基底合成モジュール２２６は、量子化解除基底係数を基底ベクトルに適用することによってカレントメッシュに近似化する。その後、加算器ユニット２２８は、基底合成モジュール２２６からの近似メッシュを量子化解除残余と結合して、再構築メッシュを計算する。

【０１２５】

量子化（Quantization）

符号化時間変化ジオメトリのコンポーネントを表している数を量子化するために使用できる、従来の量子化手法にはさまざまなものがある。ワープ(warp)係数と残余を量子化するために、圧縮器は３レンジ正規量子化器を使用している。参考文献としては、Allen Gersho、Robert M. Gray共著「ベクトル量子化と信号圧縮(Vector Quantization and Signal Compression)」(Kluwer Academic Publishers, 1991)がある。メインレンジは平均の標準偏差内の信号に対するもので、他の２つは下位と上位のアウトライア(outlier)に対するものである。第１パスでは、信号の統計が計算され、第２パスでは、実際の量子化が行われる。

【０１２６】

空間時間詳細レベル（Spacetime Level of Detail）

メッシュ単純化手法は、3D時間依存ジオメトリをメッシュ精細化（mesh refinements）の階層に変換するために使用することができる。メッシュ単純化手法の例をいくつか挙げると、Hugues Hoppe著「プログレシップメッシュ(Progressive

Meshes)」、pp. 99-108, SIGGRAPH 95およびHugues Hoppe著「プログレッシブメッシュのビュー依存精細化(View-Dependent Refinement of Progressive Meshes)」、pp. 189-198, SIGGRRAPH 97に記載されているプログレッシブメッシュがある。プログレッシブメッシュを詳しく説明した文献としては、次の米国特許出願、すなわち、第08/586,593号、発明の名称「プログレッシブメッシュの符号化とプログレッシブ伝送(Encoding and Progressive Transmission of Progressive Meshes)」(Hugues Hoppe)、第08/797,502号、発明の名称「プログレッシブメッシュのメッシュ単純化と構築(Mesh Simplification and Construction of Progressive Meshes)」(Hugues Hoppe)、第08/7907,501号、発明の名称「プログレッシブメッシュのジオモルフと可変解像度制御(Geomorphs and Variable Resolution Control of Progressive Meshes)」(Hugues Hoppe)、および第08/797,781号、発明の名称「プログレッシブメッシュの選択的精細化(Selective Refinement of Progressive Meshes)」(Hugues Hoppe)がある。なお、これらの内容全体は引用により本明細書に含まれている。

【0127】

二次曲面誤差メッシュ単純化(quadric error mesh simplification)は、Michael Garland、Paul S. Heckbert共著「二次曲面誤差メトリックスを使用した表面単純化(Surface Simplification Using Quadric Error Metrics)」、pp. 209-216, SIGGRAPH 97に記載されている。

【0128】

MetaCreations Corporation提供のMetaStream 3Dファイルフォーマットで使用されているメッシュ単純化のように、他のメッシュ単純化手法を使用することも可能である。

【0129】

時間依存ジオメトリの圧縮で詳細レベルコントロールを使用すると、いくつかの利点が得られる。詳細レベルコントロールを使用すると、3Dジオメトリを表すメッシュが単純化されるので、時間依存ジオメトリの符号化コストが大幅に低減される。単純化されたジオメトリの動きは詳細化メッシュに表れた動きによく近似しているので、ジオメトリの単純化バージョンを使用すると、メッシュ変換コ

ーダではアフィン変換、基底分解コーダでは基底ベクトルといったように、圧縮パラメータを計算することができる。

【0130】

詳細レベルコントロールのもう1つの利点は、このコントロールによると、圧縮器が時間を通るメッシュのトポロジを変更できることである。特定の3Dオブジェクトがシーンのどこに置かれているか、そのオブジェクトがどれだけ変化するかに応じて、オブジェクトの時間依存ジオメトリは、高詳細レベルでも、低詳細レベルでも表現することができる。従って、3Dオブジェクトまたはオブジェクトの一部は、選択した時点でのアニメーションシーケンスの中でどれだけ重要であるかに応じて、変化する詳細レベルで表現することができる。

【0131】

時間と空間の両方でメッシュ単純化を拡張すると、時間依存ジオメトリの効率的な表現が得られる。具体的には、メッシュ単純化は、時間依存ジオメトリを空間と時間でピラミッドとして表現するデータ構造を作るように拡張することができる。このピラミッドは、時間と空間の両方で3Dオブジェクトの詳細レベルの階層を表している。この空間－時間ピラミッドが一種の圧縮として使用できるのは、オブジェクトの時間依存ジオメトリの階層表現が、アニメーション内のフレームごとにメッシュからなるオリジナル時間依存ジオメトリよりも小さくなるからである。また、この階層表現が伝送するのに効率的であるのは、階層表現を精細化（refine）または粗粒化（coarsen）するために使用される精細化と粗粒化レコードが、フレームごとにメッシュを伝送する代わりに伝送できるからである。上述したように、階層表現によると、上述した他の圧縮方式は、空間－時間ピラミッドの該当詳細レベルで動作するように設計できるので、さらに効率化されることになる。

【0132】

ローカルフレーム（Local Frames）

以下で詳しく説明するように、空間－時間ピラミッドの階層表現は、一連の拡張レコードを使用して表現することができる。メッシュ表面のローカル座標で拡張レコードを符号化すると、動きの多くはファイン詳細化することができる。こ

のローカル座標には、メッシュ形状の階層コントロールや階層量子化といった、いくつかの利点がある（そこでは、ファイン詳細を符号化するビット数が少なくなる）。これらの利点は細分化方式（subdivision scheme）で得られる利点と類似している。細分化方式に関する参考文献としては、Denis Zorin、Peter Schroeder、Wim Sweldens共著「任意トポロジをもつメッシュの補間細分化(Interpolating Subdivision for Meshes with Arbitrary Topology)」、pp. 189-192, SIGGRAPH 96、Denis Zorin、Peter Schroeder、Wim Sweldens共著「インタラクティブマルチレゾルーションメッシュ編集(Interactive Multiresolution Mesh Editing)」、pp. 259-268, SIGGRAPH 97、およびMichael Lounsbery、Tony D. DeRose、Joe Warren共著「任意トポロジカルタイプの表面のマルチレゾルーション分析(Multiresolution analysis for Surfaces of Arbitrary Topological Type)」、pp. 34-73, ACM Transaction on Graphics, volume 16, January 1997がある。しかし、時間依存ジオメトリという意味では、ローカル座標系のコヒーレンスは時間を通すとさらに大きくなる。

【0133】

本実施形態では、空間－時間ピラミッドの階層は、時間依存頂点位置のマトリックスを、1) 縮小したジオメトリの集合と、2) 拡張レコードに因数分解することにより作られている。次の式は、頂点位置のマトリックスが、どのように縮小頂点列のマトリックスと頂点拡張に因数分解されるかを示している。

【0134】

【数22】

$$V_E = \left[\begin{array}{ccccc} v_0^0 \cdots v_0^{j_1-1} & v_0^{j_1} \cdots v_0^{j_2-1} & \cdots & v_0^{j_{n-1}} \cdots v_0^{m-1} & \\ \vdots & & & \vdots & D \\ v_{m-1}^0 \cdots v_{m-1}^{j_1-1} & v_{m-1}^{j_1} \cdots v_{m-1}^{j_2-1} & \cdots & v_{m-1}^{j_{n-1}} \cdots v_{m-1}^{m-1} & \end{array}\right]$$

縮小頂点列　　頂点拡張

【0135】

縮小頂点列の各々は、空間階層におけるペアの隣接レベル間のデルタを表して

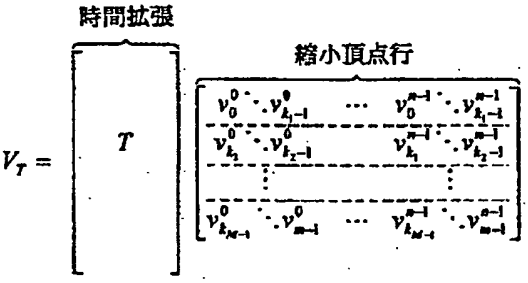
いる。頂点拡張には、各々の対応する縮小頂点列の拡張レコードが含まれ、縮小頂点列をどのように拡張すると、オリジナルマトリックスの近似に復元できるかを示している。

【0136】

マトリックス位置の頂点は、次式に示すように、時間拡張と縮小頂点行のマトリックスに因数分解することも可能である。

【0137】

【数23】

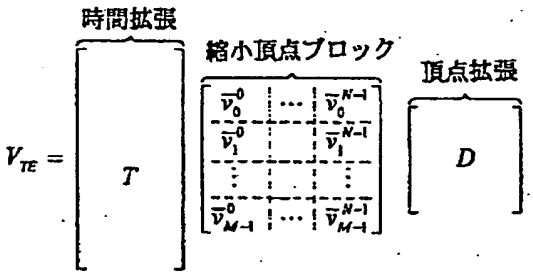


【0138】

上式において、行は、時間次元における階層の隣接ペア間のデルタを表している。時間拡張Tは、各縮小頂点行に対応する時間拡張レコードを含み、対応する行をどのように拡張すると、オリジナルマトリックスの近似に復元できるかを定義している。前記2式の考え方を結合すると、次式に示すように、頂点位置のマトリックスは時間拡張、縮小頂点ブロックのマトリックス、および頂点拡張に因数分解することができる。

【0139】

【数24】



【0140】

縮小頂点ブロックは、空間－時間階層における隣接列と行の間のデルタを表している。

【0141】

精細化基底コード化 (Refinement Basis Coding)

本実施形態では、メッシュ精細化手法を使用して一連のエッジ縮小/頂点分割レコード (edge-contraction/vertex-split record) を作成している。メッシュの詳細レベルを減少するには、頂点位置を相互接続するエッジは、頂点が少なくなったメッシュを作成するように縮小される。逆に、メッシュの詳細レベルを増加するには、頂点位置は、詳細化されたメッシュを作成するように追加位置に分割される。この実装では、デルタ値の集合をメッシュ精細化のトポロジから分離することによって、次の形式の精細化マトリックスを得ている。

【0142】

【表1】

$Split_0$		$Split_1$			$Split_{n-2}$		$Split_{n-1}$	
$D_{0,0}$	$D_{0,1}$	$D_{0,2}$	$D_{0,3}$	...	$D_{0,2n-4}$	$D_{0,2n-3}$	$D_{0,2n-2}$	$D_{0,2n-1}$
...								
$D_{m-1,0}$	$D_{m-1,1}$	$D_{m-1,2}$	$D_{m-1,3}$	...	$D_{m-1,2n-4}$	$D_{m-1,2n-3}$	$D_{m-1,2n-2}$	$D_{m-1,2n-1}$

【0143】

上のマトリックスは頂点拡張リストを表している。マトリックスの上段行は時間依存ジオメトリの空間階層の分割を示している。マトリックス内の後続行は時間のインクリメントに対応している。マトリックスの列は空間ドメインにおける階層の隣接レベル間のデルタ値に対応している。

【0144】

本実施形態では、縮小係数は右から左に向かい、逆に拡張係数は左から右に向かっている。精細化プロシージャの構造により、マトリックスの各要素にストア

されたデルタベクトルの大きさは、左から右に向かって減少している。当然に理解されるように、これは、縮小される最初のエッジがオリジナルメッシュを乱すことが最小になる設計になっているためである。信号強度はマトリックスの左側が強く、マトリックスの左から右に向かって徐々に小さくなっている。

【0145】

精細化マトリックスを使用すると、上述した圧縮方法、具体的には、ジオメトリック変換方法と基底伸張方法の効率が向上する。例えば、テストケースでは、メッシュ精細化方法の本実施形態によれば、約3000の頂点をもつオリジナルメッシュは精細化マトリックスに変換されている。プロトタイプのジオメトリック変換コーダで計算されたアフィン変換係数とプロトタイプの基底分解コーダで計算された基底ベクトルは、精細化マトリックスの左側で300 − 1000要素を使用するだけで扱いやすくなっている。事実、このメッシュ単純化を行うと、メッシュがローパスフィルタの働きをするという利点が得られる。

【0146】

拡張リストで要素をどのように配列するかは、メッシュ単純化プロセス期間に使用される最適化基準によって決定される。本実施形態では、アニメーションの第1フレームのメッシュで二次曲面誤差測定プログレッシブメッシュ手法（quadric-error measure Progressive Mesh technique）が使用され、その結果がジオメトリデータのオリジナル時間依存マトリックス内の行の残りに適用されている。この方法は十分に高速で、正確である。しかし、圧縮をダイナミックに行うという意味では、このように配列を行うことは、アニメーションシーケンス全体では最適でない。もっと最適な配列を見つけるには、頂点分割の依存性グラフを使用すると、二次曲面誤差測定のように、同時に使用される圧縮と忠実度測定によって拡張レコードを再配列することができる。言い換えれば、どの分割列が再配列できるか、マトリックスのさらに左側の頂点が分割に依存するためどの分割列が再配列できないかは、頂点分割の依存性グラフによって決まることになる。もっと一般化された手法では、最良の圧縮/最良のビジュアル品質が得られる特定のプログレッシブメッシュ（PM）拡張が探索されるようになっている。特定の瞬時では（言い換えれば、Vのある行が与えられているとき）、このプログレッシ

ブメッシュ手法では、低詳細レベルメッシュの空間ジオメトリをオリジナルメッシュの空間ジオメトリに合致させる、最良の頂点縮小のシーケンスが見つけられる。時間依存ジオメトリでは、特定のPMの品質を測定するためには、PMの詳細レベルごとに時間次元にわたって（つまり、オリジナル頂点マトリックスVの行にわたって）ジオメトリの誤差を加算する必要がある。

【0147】

例えば、PMはVの行ごとに計算したあと、Vの全行にわたってテストして、ジオメトリの正確性を確かめることができる。また、Vの列の平均をとることによって得られた平均メッシュに対してPMを計算することもできる。

【0148】

空間－時間コーダのオペレーションを示す例

時空コーダの理解を容易にするために、以下では、人物を例にして説明することにする。ある距離からは、必要になることは、手足を固定させたまま、ボディ全体を低ポリゴンカウントで並行移動し、回転させることである。人物が近づくと、動きの正しい知覚を伝達するために手足のおおまかな運動と形状が必要になる。人物がさらに近づくと、顔と他の筋肉の動きと形状のファイン詳細が大きくなる。その後、人物が後退すると、上に挙げたステップとは逆順に、必要になる詳細は小さくなる。詳細を段階的に大きくした後、小さくしていくために、空間－時間コーダは、最初に、モデルの単純なジオメトリック表現とアニメーションの単純なバージョンを送信する。その後、クライアントのビューポイントに基づいて、サーバは、空間的および時間的詳細更新を必要に応じてストリームダウンする。

【0149】

ここで言うサーバとは、時間と共に変化するジオメトリのソースの働きをするシステムエンティティのことであり、クライアントとは、そこでプレイバックが行われるエンティティのことである。ここで言う広い意味では、必要になる帯域幅とメモリを低減化するアプリケーションは広範囲にわたっている。サーバとクライアントは、同一または異なるプロセッサ上で実行されるソフトウェアプロセスにすることができる。サーバは、ホストプロセッサ上で実行されるソフトウェ

アモジュールにすることができ、クライアントは、コンピュータシステムのグラフィックスコプロセッサにすることができる。サーバは第1コンピュータ上で実行されるソフトウェアにすることができ、クライアントは第2コンピュータで実行されるハードウェアデバイスまたはソフトウェアモジュールにすることができ、この場合、第2コンピュータは、モデム接続、ローカルエリアネットワークまたは広域ネットワークを介して第1コンピュータに接続されている。

【0150】

空間－時間ピラミッドの導出

入力は、高密にタイムサンプリングされたメッシュM(0), ..., M(n-1) の集合である。これまでのセクションで上述したように、頂点の位置はマトリックスVとして考えることができる。Vの各行iはシングルタイムサンプルVi = M(i) に対応している。Vの各列j は時間を通る座標の軌跡である。空間－時間ピラミッドは空間（列にまたがる）と時間（行にまたがる）の2つの次元に沿ってVを分解したものである。これを具体的に説明するために、左から右へ移動する立方体の単純な1秒アニメーションについて考えてみる。そこでは、立方体の上部は底部に対して垂直軸を中心に周期的に回転（ツイスト）している。このアニメーションは図7に示されている。立方体の上部と底部は、x － z 平面に同一線上の四角のままになっているが、立方体の側辺は、図7の上段アニメーションシーケンスに示すように、上部のy回転が増減したとき、曲面になっている。このアニメーションは60Hzでタイムサンプリングされ、60個の初期メッシュM(0), ..., M(59) を得ている。上部頂点はV0からV3までであり、下部頂点はV4からV7までである。頂点の残りは立方体の回転している側面を符号化している。

【0151】

最初のステップでは、メッシュが係数にセグメント化され、これらの係数は独立にアニメーション化された後、アニメーション式と結合される。立方体では、望ましいアニメーション式は次のようになっている。

【0152】

【数25】

```
Translation * (Cube + YRotation(k) * CubeY(k)).   (A)
```

ただし

```
YRotation(k) = AxisRotation(YAxis, Interp(k, alpha(k) *
TopTheta)).        (B)
```

【０１５３】

k項の各々は、異なる回転速度でメッシュのセグメントを符号化している。下部セグメントは立方体の下部と合致し、静止している。下から上へ移動するとき、各セグメントの速度は、立方体の上部のY回転と合致する上部セグメントに至るまで徐々に増加している。

【０１５４】

生の入力シーケンスからこのアニメーション式を得るために、ジオメトリック変換符号化ステップのリストがメッシュに適用される。例示のリストはカレントVマトリックスの平均値軌跡であり、その後、一連のアフィン合致ジオメトリック符号化になっている。Vマトリックスメッシュの各行の平均を求めると、マス中心の軌跡が得られ、それからアニメーション式のTranslation項が得られる。立方体の並行移動は、図７の下段アニメーションシーケンスに示されている。

【０１５５】

次に、一連のアフィン合致ジオメトリック変換符号化はY軸を中心に回転している頂点の残りと合致される。頂点マトリックス全体に合致されるY回転項が立方体の中心に良好に合致しているのは、立方体の中心近くの頂点が平均的回転と共に移動するからである。符号化はY軸の中心から始まり、その後、アニメーション式の、次の回復項にどれだけ合致しているかに基づいて、上部までと下部まで外側に移動していく。これから得られるアニメーション式は、次のような形式になっている。

【０１５６】

【数２６】

```
Translation * (Cube + YRotation(0) * (CubeY(0) + YRotation(1) *
(CubeY(1) + .... YRotation(N-1) * CubeY(N-1)).   (C)
```

【０１５７】

単純な代数で単純化すると、これは所望のアニメーション式(A)になる。

【0158】

詳細項は、次の2パラグラフで説明するように、空間的または時間的に符号化することができる。エンコーダの仕事は、符号化シーケンスの各ステップで、最大のコヒーレンスをもつ次元を選択することである。これは、次元を利用すると共に、最良の予測で詳細スムーズ化を選択することによって行われる。

【0159】

各セグメントの空間的次元は、代表的なメッシュを使用するプログレッシブメッシュのエッジ縮小（edge collapse）を使用してコード化され、この代表的なメッシュはシーケンスの最初のものであるか、初期の未アニメーション化メッシュ（静止形状とも呼ばれる）であるか、のどちらかである。エッジ縮小を行うと、Vの2列が結合されて1つの列になり、オリジナル列を回復するためにデルタが符号化される。

【0160】

各セグメントの時間的次元をコード化するには、2つの方法がある。スパースでスムーズなデータでは、頂点位置とアニメーション係数は時間を通るようにスプライン（spline）される。より高詳細レベルは、2つの既存頂点の間に新しいスプライン頂点を挿入することにより追加され、動きは新しいローカル特徴をもつことになる。時間エッジ縮小を行うと、カレントセグメントの2行が結合され、オリジナル行を回復するために行デルタが符号化される。高密度アニメーションでは、ウェーブレット（wavelet）が使用され、より高詳細レベルがより多くの詳細係数と共に追加される。詳細レベル間でスムーズに移行するために、クライアント側の表現は、よりスムーズな係数を残したまま、新しい詳細係数とブレンドされる。

【0161】

ビューに依存する、望ましい空間的および時間的解像度が与えられているとき、チェックポイントがセーブされ、符号化は各次元で反復的に開始される。これにより、メッシュのピラミッド格子が得られる。格子の各ノード間のリンクは、一方から他方へ移動するために必要な一連の更新である。図8はピラミッドの1

レベルを示している。この構造が繰り返されて、完全なピラミッドが得られる。

【0162】

空間－時間コーダで使用されるデータ構造の例は以下に示されている。データ構造は、それがサーバに置かれているか、クライアントに置かれているか、あるいは3Dアニメーションシーケンスの選択的プレイバック期間にクライアントとサーバの間で送信されるかどうかに基づいて分類されている。

【0163】

クライアント側データ構造のフォーマット

次のデータ構造は、クライアント側でアニメーションを表現するために使用されるものである。

【0164】

【表2】

```
//パラメータ空間の単一点のトライアングルメッシュ
struct Mesh {
    int m_nvertex;      //頂点の数
    int m_ndim;         //頂点次元の数（x,y,z+他）
    float *m_avertex;   // nvertex  *ndim頂点の配列
    int m_nitri;        //トライアングルの数
    int (*m_aitri)[3];  //m_avに入り込むトライアングルインデックス
};

//1次元スプライン
struct SplineID {
    int m_nsample;      //タイムサンプルの数
    struct sampleID {
        float m_t;      //時間値
        float m_av[0]   //ndim値の配列
    } m_asample[0]      //可変間隔サンプルの配列
};

//アフィン変換セグメント
struct AffineVertexSegment {
    //ギャザ／スキャッタ（gather/scatter）次元には、頂点インデックスと頂点値の２
つがある
    int m_nvertex;      //<= nvertex
    int *m_aivertex;    //影響を受ける頂点のインデックス
    int m_ndim;         //<= 頂点の ndim
    int *m_aidim;       //影響を受ける次元のインデックス
    float *m_av;        // nvertex *ndim静止形状の値
    SplineID *m_xfm;    //ndim *(ndim+1)アフィン変換のスプライン係数
    SplineID *m_avs;    // nvertex *ndimスプライン残余
};

//アフィン変換メッシュ
struct MeshAffineID {
    Mesh m_mesh;        //カレントメッシュ
    int m_nsegment;     //アフィン変換セグメントの数
    AffineVertexSegment *m_avs;    //アフィン変換セグメントのリスト
};
```

【０１６５】

サーバからクライアントへのデータ構造のフォーマット

サーバは次のデータ構造をクライアントに伝達し、選択的プレイバック期間にクライアント側でアニメーションを更新することを可能にする。

【０１６６】

【表３－１】

```
/*
//更新レコードのストリームのテクスチャ表現

//持続性更新
mesh imesh nvertex ndim          //メッシュのインデックスとサイズ
segiv imesh iseg seg_nvertex iv(0) ... iv(seg_nvertex-1)   //アクティブ頂点のリスト
segdim imesh iseg seg_ndim idim(0) ... idim(seg_ndim-1)   //アクティブ次元のリスト
triset imesh itri iv0 iv1 iv2      //トライアングル頂点インデックス
tridel imesh itri            //インデックス付きトライアングルを削除

//頂点初期化と削除
vset imesh iv v(0) v(1) ... v(ndim-1)   //頂点位置、法線、uvなどを初期化
voffset imesh iv ivsrc dv(0) dv(1) ... dv(ndim-1)   //先行頂点から頂点を初期化
vdel imesh iv                  //インデックス付き頂点を削除

//セグメント位置/時間更新
segxfm imesh iseg t xfm(0) ... xfm(seg_ndim *(seg_ndim+1) -1)
segresid imesh iseg iv t dv(0) ... dv(seg_ndim-1)
```

【０１６７】

【表３－２】

```
*/
//更新構造
struct UpdateTriangle {
        int m_itriangle;        //追加/更新するトライアングルのインデックス
        int m_aivertex[3];      //頂点インデックスのリスト
};

struct UpdateDeleteTriangle {
        int m_itriangle;
};

struct UpdateSetVertex {
        int m_ivertex;          //追加/更新する頂点のインデックス
        float m_afvertex[0];    //サイズndim頂点値の配列
};

struct UpdateOffsetVertex {
        int m_ivertex;      //追加/オフセットする頂点のインデックス
        int m_ivertexSource; //ソース頂点のインデックス
        float m_adfvertex[0]; //サイズndimデルタ頂点値の配列
};

struct UpdateDeleteVertex {
        int m_ivertex;      //削除する頂点のインデックス
};
```

【０１６８】

【表３－３】

```
struct UpdateSegmentActiveVertices {
      int m_isegment;      //追加/更新するセグメントのインデックス
      int m_nvertex;   //リスト内のアクティブ頂点の数
      int m_aivertex[0];     //アクティブ頂点インデックスのリスト
};
struct UpdateSegmentActiveDimensions {
     int m_isegment;       //更新するセグメントのインデックス
     int m_ndim;           //アクティブ次元の数
     int m_aidim[0];       //アクティブ次元のリスト
};

struct UpdateSegmentxfm {
     int m_isegment;       //更新するセグメントのインデックス
     float m_t;            //パラメータ位置
     float m_afxfm[0];     //サイズseg_ndim*(seg_ndim+1)xfm値の配列
};

struct UpdateSegmentResidual {
     int m_isegment;       //更新するセグメントのインデックス
     int m_ivertex;        //セグメント内の頂点のインデックス
     float m_t;            //パラメータ位置
     float m_afvertex[0];  //size seg_ndim頂点値の配列
};

//すべての更新構造のタイプ結合
struct Update {
      int m_cb;     //更新レコード全体のサイズ
      int m_type;   //更新のタイプ
      union {
            UpdateTriangle triset;
            UpdateDeleteTriangle tridel;
            UpdateSetVertex vset;
            UpdateOffsetVertex voffset;
            UpdateDeleteVertex vdel;
            UpdateSegmentActiveVertices segiv;
            UpdateSegmentActiveDimensions segdim;
            UpdateSegmentxfm segxfm;
            UpdateSegmentResidual segresid;
     };
};
```

【０１６９】

サーバ側データ構造のフォーマット

次のデータ構造は、クライアント側でのアニメーションの選択的プレイバックに関連してサーバ側に置かれているものである。

【０１７０】

【表４】

```
struct UpdateBlock {
        //次の詳細レベルに到達するために移動する方向
#define TIME_FINE       0
#define TIME_COARSE     1
#define TIME_NEXT       2
#define TIME_PREV       3
#define SPACE_FINE      4
#define SPACE_COARSE    5
#define SPACE_X         6
#define SPACE_Y         7
#define SPACE_Z         8
        UpdateBlock *m_pStep[9];   //空間的と時間的の精細化と粗粒化詳細レベルに対応する更新レコードのブロック
        int m_nupdate;          //このブロック内の更新レコードの数
        Update **m_apupdate;    //更新レコードを指すポインタの配列
};

struct ServerMeshAffineID {
        MeshAffineID m_meshClient;    //サーバは、クライアントが該当更新シーケンスを計算するためになにをもっているかを記録にとっている
        UpdateBlock *m_pupdateBlock;      //時空ピラミッド内のカレント位置
};
```

【０１７１】

アニメーションの伝送期間中、サーバとクライアントの間では次のような通信が行われる。

【０１７２】

【表５】

サーバ ← クライアント	空間的解像度、カレントビュー
サーバ → クライアント	空間的更新レコード（例：プログレッシブメッシュスタイルの頂点分割とエッジ縮小）

サーバ ← クライアント	時間的解像度、カレント時間インターバル
サーバ → クライアント	時間的更新レコード（パラメータ曲線結び目挿入と削除、またはウェーブレット詳細係数のブロック）

【０１７３】

クライアントは、セグメントのカレントビューポイントを、所望の空間的解像度（spatial resolution）と一緒にサーバに送信する。空間的解像度は、クライアントがセグメントをレンダリングするときのジオメトリック詳細レベルを示している。これに応答して、サーバは空間的更新レコードを返送し、クライアントが空間－時間ピラミッドから所望のジオメトリック詳細レベルを抽出できるようにする。

【０１７４】

セグメントが更新されるときのレートを指定するために、クライアントは時間的解像度（temporal resolution）とカレント時間インターバルをサーバに送信する。これに応答して、サーバは時間的更新レコードを送信し、クライアントが空間－時間ピラミッドから所望の時間的詳細レベルを抽出できるようにする。

【０１７５】

他の次元にまたがる符号化

これまでは、本明細書では代表的な次元として時間を使用してきた。リニアシーケンスをプレイバックするだけでなく、この次元はインタラクティブコントロールのためにも使用できる。例えば、ひじを曲げる詳細アニメーション（隆起す

る筋肉、スライドするスキンなどと共に）は、ひじのジョイントアングル（関節角度）によってパラメータ化されている。ユーザがジョイントアングルを指定し、アニメーションシーケンスをその指定に応じて伸張（decompress）できるようにすると、ユーザは高詳細のアニメーションを制御することができる。これは、ゲームなどのインタラクティブアプリケーションで利用すると便利である。

【0176】

この特定例では、曲げるひじのアニメーションは、空間—角度マトリックス構造で表現することができる。各列はひじのオブジェクトの3Dメッシュ内の位置を表し、行はひじの関節の角度位置を表している。時間と同じように、関節角度は、低詳細レベル（例えば、関節角度サンプルが少ない）から高詳細レベル（例えば、関節角度位置を多くしてひじの曲がりを詳細に表している）までの階層構造で表現することができる。選択的プレイバック期間、クライアントは角度の動きに見合った詳細レベルを要求し、この要求に応答して、サーバは関節角度次元の、対応する更新レコード（粗粒化または精細化のどちらか）を送信し、ひじの運動が該当詳細レベルでアニメーション化されるようにする。例えば、ユーザはひじの関節の角度位置を、ジョイスティックなどの入力デバイスから指定することができる。この場合、クライアントは角度位置を計算し、それをサーバに送信すると、サーバからは、該当更新レコードが戻される。

【0177】

類似の手法を使用して、他の次元（例えば、ある軸を中心とする回転運動、軸に沿う並行運動、カーブした通路上の動きなど）上の3Dアニメーションを符号化することもできる。

【0178】

テクスチャとジオメトリの結合圧縮

頂点位置のほかに、各頂点は、テクスチャ座標のように、3D頂点位置に関して上述した同じ手法を用いて圧縮することができる、他のデートと関連付けることも可能である。テクスチャマップオペレーション（texture map operation）では、グラフィックスレンダリングシステムは2D画像を3Dオブジェクトの表面にマッピングする。オブジェクトの各頂点のテクスチャ座標は、2Dテクスチャ空間に

おける、その頂点の対応する位置を表している。出力画像のピクセルを計算するとき、グラフィックスレンダリングシステムはテクスチャ座標を使用して、テクスチャ画像内の該当テクスチャサンプル（1つまたは複数）をピクセルごとに見つける。テクスチャマップオペレーションの形態に応じて、グラフィックスレンダリングシステムはテクスチャサンプルをフィルタに通してから、その結果のカラー値を出力ピクセルに適用することができる。テクスチャ座標は、テクスチャ座標が3D空間ではなく2D空間の位置を表していることを除けば、3D頂点位置のメッシュと同じマトリックス形式で表現することができる。構造が類似しているために、時間変化3Dジオメトリのマトリックスに適用される同じ圧縮手法を、テクスチャ座標のマトリックスにも適用することが可能である。

【0179】

代表例として、テクスチャ座標はスタッチク（静的）であるが、これは必須ではない。例えば、川の水のさざなみの画像は、テクスチャ座標をアニメーション化することだけで下流に移動させることができる。具体的には、特定頂点のテクスチャ座標は時間の関数として表現することができる。ビジュアル効果が大きいクラスでは、アニメーション化ジオメトリとアニメーション化テクスチャを結合することは不可欠である。テクスチャ座標を変化させる新しいアプリケーションは、アニメーション化ジオメトリ圧縮の基礎が与えられていれば、実現することができる。

【0180】

詳細レベルの表現は、ジオメトリ座標とテクスチャ画像の両方を含んでいることが好ましい。ジオメトリック詳細レベルが低であれば、テクスチャ座標には対応する低詳細レベルを使用するだけで十分である。テクスチャ画像の空間的解像度は、空間的解像度レベルが変化する階層方式で、可能ならば、空間－時間ピラミッドにおけるジオメトリック詳細レベルに対応付けて、符号化することができる。選択的プレイバックのコンテキストでは、サーバは最初にテクスチャ画像の低詳細レベル係数を送信し、その後、必要に応じて高詳細レベルを送信すれば、よりファインな詳細を必要とするビューポイントでテクスチャ画像を精細化することができる。画像とジオメトリの両方の更新を結合すると、与えられた解像度

でどちらが重要であるかに応じて、トレードオフを行うことができる。アニメーション化テクスチャ画像では、このトレードオフは時間的詳細と空間的詳細の両方で行うことができる。

【０１８１】

時間的および空間的符号化の対象となる高次表面

高密度にサンプリングされた頂点マトリックスVの初期表現では、２つの補間マトリックスS（時間を通る）とG（空間を通る）は、連続する位置マトリックスP＝ＳＶＧを得るようにグラフィックスシステムによって定義されるものと想定していた。しかし、SとGが初期に一定補間または線形的補間として与えられていれば、特に最新のグラフィックスハードウェアには、スプラインや細分化表面のような、高次ジオメトリック補間のサポートが含まれているので、高次補間値(interpolator) S' とG' をVから導き出すようにすると好都合である。

【０１８２】

最良S' とG' の推定は下から上に向かって行われる。最初に、時間的および空間的に小さな近隣値がランダムに選択され、与えられた次数のローカル最良合致スプライン表面と適合する。次に、最良合致をもつ近隣値の成長と合体（coalesce）を繰り返すと、S' とG' のローカル貪欲最適値（local greedy optimum）が見つかる。これは、前述したトライアングルセグメンテーションに似ているが、空間的座標と時間的座標の両方を使用して、合致領域の規模を決定している。

【０１８３】

高次補間値が正確に合致しないときでも、これらは、表面が時間的にどのように移動するか、あるいは空間的に点から点へどのように変化するかを予測する手段として使用することができる。残余R＝ＳＶＧ－ S' V' G' は、オリジナルマトリックスP＝S' V' G' ＋Rを回復するために使用できる。

【０１８４】

残余の符号化のための変位マップ

大部分のグラフィックスハードウェアは、2Dカラー画像を形状の表面ジオメトリに効率良くマッピングするテクスチャマッピングハードウェアを含んでいる。最新グラフィックスハードウェアは「バンプ(bump)」マッピングも含んでおり、

そこでは、2D画像はカラーではなく、形状の法線（normal）を乱し、関心のある照明効果が得られるようにしている。いくつかの最新グラフィックスハードウェアは「変位(displacement)」マッピングも含んでおり、そこでは、オフセットの2D画像が形状の位置を乱すようにしている。

【0185】

時間依存ジオメトリコーダの残余を表面のローカル座標に変換すると、変位マッピングは残余を伸張された表面に適用するために使用することができる。時間依存残余は、変位マップとして符号化された複数の残余をブレンドすることで計算することができる。グラフィックスシステムは高速データ通路と高速頂点プロセッサを装備しているので、このグラフィックスシステムは、最終的残余補正を伸張された形状に適用する効率的な場所になっている。これが特に効率的な手法となるのは、前述した高次表面符号化と併用したときである。

【0186】

実施形態の例

本実施形態では、上述した圧縮方法を組み合わせたものを使用して、時間依存ジオメトリを表す頂点位置のマトリックスを圧縮している。図9はこの実施形態におけるコンピュータを示すブロック図であり、図10は伸張器を示すブロック図である。

【0187】

図9に示すように、頂点位置のマトリックスVで表されている、時間依存ジオメトリのストリームを圧縮するプロセスは時間依存メッシュを単純化することから始められる。本実施形態では、アニメーションの第1フレームのメッシュはベースメッシュとして使用されている。しかし、上述したように、ベースメッシュを決定するには、さまざまな代替方法があり、この中には、モデルの作成者が事前定義したもの、マトリックスVに含まれる一連のメッシュの平均から導き出されるもの、などが含まれている。本実施形態では、アニメーションの第1フレームのメッシュに二次曲面誤差測定手法を使用して、ペア縮小の集合を上述した精細化マトリックステーブル行の1つの形で得ている。図9に示す単純化メッシュブロック250は、頂点位置のメッシュについてGarlandの二次曲面誤差測定手

法を実装するために使用されるルーチンを表している。

【0188】

結果として得られるペア縮小の集合は、オリジナル時間依存頂点マトリックスVの行の残りに適用され、ブロック252に示すような精細化マトリックスが得られる。この精細化マトリックスは拡張レコードを通るように「順方向にローリング」され、マトリックスVに含まれるオリジナルメッシュの各々の単純化バージョンが作成される。

【0189】

ブロック252からの単純化メッシュは、単純化ベースメッシュを含めて、ベースメッシュとマトリックスV内の他のメッシュとの間の最良アフィン合致を見つけるために使用される。マトリックスV内のメッシュの各々は、アニメーションシーケンス内のタイムサンプル、つまり、アニメーション内のフレームに対応している。変換合致ブロック254は、各フレームのアフィン変換係数の集合を計算するために使用されるルーチンを表している。これらのルーチンは低詳細レベルの頂点マトリックスを使用して、

【0190】

【数27】

$$A_K\hat{V}_0 = V_K$$

【0191】

に対する最小2乗解の近似値を計算する。変換ブロック254には、正規方程式の方法を実行して変換係数の解を求めるルーチンも含まれており、そこには、ベースメッシュの4×4適合マトリックスの累算と反転、およびn×4マトリックスと適合マトリックスとのマトリックス積が次のように含まれている。

【0192】

【数28】

$$K = V_0^T(V_0V_0^T)^{-1}$$
$$A = VK$$

【0193】

本実施形態では、頂点位置の時間依存マトリックスVにおけるメッシュの各々は、そのローカル座標系に表現されている。このようにすると、時間を通るメッシュの動きのコヒーレンスが向上するが、圧縮器の計算複雑性も増加することになる。精細化マトリックスのメッシュはローカル座標に表現されるので、ローカル座標はグローバル座標に変換される必要がある（言い換えれば、メッシュが変換される必要がある）。本実施形態では、これは、アフィン変換係数が計算された後、新しい座標内のメッシュの各々を単純化し直すことによって解決されている。

【0194】

精細化マトリックスを新しい座標系に変換する別の方法は、拡張レコードに入っているデルタ値を、それぞれのローカル座標系から新しいグローバル座標系に変換することである。図9のブロック256は、ブロック250と252のメッシュ単純化ルーチンを新座標系のメッシュの各々に適用するプロセスを表している。

【0195】

次に、圧縮器は、メッシュの各々と変換ベースメッシュとの間の差分で、各メッシュに対応するアフィン変換係数を使用して計算された差分として残余を圧縮する。アフィン変換係数と単純化ベースメッシュは共に量子化され、その後、量子化器ブロック260、262と量子化解除器ブロック264、266に示すように量子化解除される。ジオメトリック変換ブロック270は、カレントメッシュのアフィン変換係数をベースメッシュに適用して、変換ベースメッシュを計算するルーチンを表している。変換ベースメッシュと対応する単純化メッシュとの差分は図示のように減算器ブロック272によって計算される。

【0196】

時間依存ジオメトリをさらに圧縮するために、本実施形態によれば、基底分解コード化（basis decomposition coding）がオプションとしてサポートされている。なお、量子化残余（量子化ブロック274）、アフィン変換係数276、およびベースメッシュ278は、このステージでトランスミッタ280に直接に送ることができる。しかし、ある種のアニメーションでは、基底分解によると、

ジオメトリマトリックスをさらに圧縮することができ、この中には、残余マトリックス２８２と単純化ベースメッシュを表すマトリックスが含まれている。

【０１９７】

基底分解ブロック２８６は、マトリックスをその基底ベクトル２８８と重み２９０（係数とも呼ばれる）に分解するプロセスを表している。本実施形態では、精細化マトリックスの切り捨てバージョン（truncated version）を使用して、基底関数と重みの集合を計算している。このステップを行うかどうかはオプションであるのは、ある種の動きは分解に役立たないが、それでも精細化マトリックスの空間的および時間的コヒーレンスが大きいためである。従って、基底分解ステップをバイパスし、残余の精細化マトリックスで列/行予測を行うことが可能である。

【０１９８】

基底分解が残余の精細化マトリックスで行われる場合は、重みがブロック２９２、２９４に示すように列/行予測器に渡される。逆に、基底分解が適用されない場合は、フラット精細化階層が列/行予測器に渡される。圧縮器は、量子化ブロック２９６と２９８に示すように、正規量子化を基底ベクトルと予測器ブロックの出力に適用する。

【０１９９】

圧縮器は、時間依存ジオメトリストリームを表す精細化階層全体を符号化するが、階層全体をランタイムにデコーダに送る必要はない。その代わりに、アニメーションの中の、与えられたビューポイントのために必要とされる空間的精細化だけが送信され、更新される必要がある。同様に、時間的精細化が必要になるのは、ジオメトリの位置の変化がアニメーションの中で見える程度までででよい。トランスミッタ２８０は、時間依存ジオメトリの、どの時間的および空間的コンポーネントを送信する必要があるかを決定するように適応させることができる。例えば、トランスミッタは時間依存ジオメトリストリームで表されたモデルのスクリーン空間投影を評価して、アニメーションの中の、与えられたビューポイントのために送信する必要のある空間的および時間的精細化を決定することができる。

【０２００】

図１０は、図９の圧縮器からの符号化時間依存ジオメトリを復号化する伸張器を示すブロック図である。レシーバブロック３２０は符号化時間依存ジオメトリをデマルチプレクスする。圧縮器の動作モードに応じて、この符号化データには、量子化基底ベクトル３２２、予測および/量子化重み３２４、階層構造３２６内の量子化ベースメッシュ、量子化ジオメトリック変換パラメータ３２８、および精細化マトリックスフォーマットの量子化残余（場合によっては、予測残余）を含めることができる。なお、基底ベクトル３２２と重み３２４を使用すると、ベースメッシュだけでなく、残余も表現することができる。量子化解除ブロック３４０－３４８は、それぞれのデータタイプの各々の量子化解除を表している。

【０２０１】

圧縮器がベースメッシュまたはベースメッシュの重みと残余で予測（行または列）を使用していれば、伸張器は逆の予測３５０を行って、マトリックスを予測以前の形態に復元する。

【０２０２】

メッシュ再構築ブロック３５２は、残余の精細化マトリックスとベースメッシュを復元する。圧縮器で行われる基底合成では、k個の浮動小数点乗算累算が行われる。ここで、kは基底ベクトルの個数である。なお、k個の基底ベクトルの部分集合nだけを送信して、精細化マトリックスを圧縮することも可能である。n個の基底ベクトルは最上位ベクトルである。例えば、k個中の最初のn個のベクトル。

【０２０３】

変換ブロック３５４はカレントメッシュの量子化解除アフィン変換係数を、ベースメッシュの再構築精細化マトリックス内の対応するエントリに適用して変換ベースメッシュを計算する。加算器ブロック３５６は量子化解除残余と、変換ベースメッシュの対応する要素を結合することを表している。残余は精細化マトリックスの一部から得られる。その後、拡張ブロック３５８は再構築精細化マトリックスの該当部分を拡張して、カレントフレームの再構築メッシュを計算する。

【０２０４】

軌跡の圧縮

上述したように、3Dモデルの動きを定義するために使用されるジオメトリックデータは、3Dモデルに関連する3D参照点の一連の変形ベクトルとして表すことができる。Guenter他を参照。変形ベクトルはマトリックス形式で表現することができる。例えば、マトリックスの列は時間インターバルに対応し、行は3D参照点の変形ベクトルに対応している。このマトリックスはマトリックスを基底ベクトルと係数に分解すると、効率よくコード化することができる。係数は時間的予測を使用してコード化することができる。量子化とエントロピコード化は、基底ベクトルと係数をコード化するためにも使用できる。

【０２０５】

図１１は、変形ベクトルのマトリックスをどのようにコード化すると、ストアと送信に効率のよいフォーマットになるかを示す図である。ジオメトリックデータは変形ベクトルのマトリックス（４００）として表現されている。この特定例では、列は、アニメーションシーケンスのフレームのような時間のインクリメントに対応している。行は、対応する3D参照点の位置を定義している3Dベクトルに対応している。

【０２０６】

分解ブロック（４０２）はマトリックスを係数（４４０）と基底ベクトル（４０６）に分解するモジュールである。時間的予測ブロック（４０８）は、係数マトリックスの列間の時間的予測を行うモジュールを表している。係数と基底ベクトルは、量子化とエントロピコード化モジュール（４１０、４１２、４１４および４１６）に示すような、量子化とエントロピコード化を使用して圧縮することができる。係数の場合には、予測は係数の予測前と後に係数のマトリックスで行うことができる。ジオメトリックデータとそのデータをストアするために使用されるマトリックスの形態によっては、係数マトリックスの列と行のどちらでも予測を使用することが可能である。

【０２０７】

エントロピコード化モジュール（４１２、４１６）の出力はトランスミッタまたはハードディスクなどのストレージデバイスに送られる。ある種のアプリケー

ションでは、変形ベクトルは、可能であれば、なんらかの形態の入力に応答して計算され、送信用にコード化されている。ここで言う「トランスミッタ」とは、バス、コンピュータネットワーク、電話回線、シリアル通信リンクなどの、なんらかの形態の通信媒体上にコード化データを伝送するために使用されるシステムソフトウェアおよび/またはハードウェアのことである。圧縮ジオメトリデータがどのように転送されるかは、通信媒体によって決まる。

【0208】

変形データの即時転送を必要としない他のアプリケーションでは、変形ベクトルを圧縮すると、依然として有利である。具体的には、圧縮データに必要な記憶スペースが少なくなり、必要とするメモリ帯域幅が低減化される。

【0209】

マトリックスを基底ベクトルと係数に分解する方法には、さまざまなものがある。以下では、主要コンポーネント分析をどのように適用すると、3Dモデルの時間変化位置を表すマトリックスを圧縮できるかを、例を示して説明することにする。

【0210】

データ集合がマトリックスAとして表されていて、データのフレームiがAの列iをマップしているとすると、Aの最初の主要コンポーネントは次のようになる。

【0211】

【数29】

$$\max_{\blacksquare}(A^T u)^T(A^T u) \qquad (3)$$

【0212】

式(3)を最大限にするuは、AA^Tの最大固有値(eigenvalue)に関連する固有ベクトル(eigenvector)あり、これは最大値の値でもある。後続の主要コンポーネントは、すべての先行主要コンポーネントに直交していることを除けば、同じように定義される。つまり、$u_i^T u_j$ for $j \neq i$。主要コンポーネントはマトリックスUで表される正規直交基底集合(orthonormal basis set)となり、そこでは、Uの列は、固有値サイズ別に配列されたAの主要コンポーネントであり、最上

位主要コンポーネントはUの最初の列に置かれている。

【0213】

Aマトリックス内のデータは、次のように主要コンポーネント基底上に投影することができる。

【0214】

【数30】

$$W = U^T A \qquad (4)$$

【0215】

Wの行iは、列A_iを基底ベクトルu_i上に投影したものである。もっと正確に言うと、Wの行iのj番目の要素はオリジナルデータのフレームjを、i番目の基底ベクトル上に投影したものに対応している。Wマトリックス投影の要素は係数と呼ぶことにする。

【0216】

同様に、Aは、次のように基底集合で乗算すると、Wから正確に再構築することができる。

【0217】

【数31】

$$A = UW \qquad (5)$$

【0218】

本発明の目的上、主要コンポーネントの最も重要なプロパティは、これらが、I_2正規形の意味で再構築するときの最良線形基底集合となることである。与えられたどのマトリックスU_kの場合も（ここで、kはマトリックスの列の数および$k < \mathrm{rank}(A)$である）、次の再構築誤差

【0219】

【数32】

$$e = \left\| A - U_k U_k^T A \right\|_F^2 \qquad (6)$$

【0220】

(ここで、

【0221】

【数33】

$$\|A\|_F^2$$

【0222】

は次のように定義されているFrobenius正規形である。)

【0223】

【数34】

$$\|A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2 \qquad (7)$$

【0224】

は、U_k がAのk個の最上位主要コンポーネントを含んでいるマトリックスであれば、最小限になる。

【0225】

データ集合Aは、その対応するWとUマトリックスの要素を量子化し、これらをエントロピコード化することで圧縮することができる。圧縮データは主要コンポーネント基底ベクトルがなければ再構築できないので、Wマトリックスと Uマトリックスの両方を圧縮する必要がある。基底ベクトルでは、DCT基底のように、オリジナルデータ集合から独立して計算できる基底集合では存在しないオーバヘッドが余分に生じることになる。

【0226】

特定の構造をもたないデータシーケンスでは、得られる圧縮効率以上に基底ベクトルの余分オーバヘッドが発生するおそれがある。しかし、正規フレーム間構造をもつデータ集合では、主要コンポーネントの基底ベクトルで再構築するときの残余誤差は、他のベースよりもはるかに小さくすることができる。この残余誤差の減少は、基底ベクトルのオーバヘッドビットを埋め合わせるのに十分な大きさになる。

【0227】

主要コンポーネントは、Strang著「線形代数とその応用(Linear Algebra and its Application)」、HBJ, 1988に記載されている特異値分解 (singular value decomposition — SVD) 法を使用して計算することができる。このアルゴリズムを効率化した実装は広く利用されている。マトリックスAのSVDは次のようになっている。

【0228】

【数35】

$$A = U\Sigma V^T \qquad (8)$$

【0229】

上記において、Uの列はA A^T の固有値、対角マトリックスΣ上の特異値Σ_i はA A^T の固有値の平方根、Vの列はA A^T の固有値である。Uのi 番目の列はAのi 番目の主要コンポーネントである。Aの最初のk個の左特異値を計算することは、最初のk個の主要コンポーネントを計算するのと同じである。

【0230】

以上、あるタイプの主要コンポーネント分析を説明してきたが、他の形態の主要コンポーネント分析を使用してジオメトリデータを圧縮することも可能である。別の形態の主要コンポーネント分析は、KL変換 (Karhunen-Loeve) と呼ばれている。

【0231】

3Dマーカの変形ベクトルの圧縮

ジオメトリックデータは、顔の動きが高度に構造化されているため、上述したように長期の時間的コヒーレンスプロパティをもっている。本実施形態を使用したあるテストケースでは、ジオメトリックデータの基底ベクトルのオーバヘッドは、キャプチャされる3Dオブジェクトに182個のマーカがあったため、固定していた。このケースでは、基底ベクトルの最大数は、3つの数x, y, zが各マーカと関連付けられているので182 * 3である。基底ベクトルのオーバヘッドはアニメーションシーケンスの長さが増加すると共に着実に減少している。

【0232】

ジオメトリックデータは、i番目のフレームの3Dオフセットデータを受け取り、それをデータマトリックスA_0のi番目の列にマッピングすることによってマトリックス形式にマッピングされる。投影係数はマトリックスM_0にストアされる。

【0233】

投影係数の列の間に顕著な相関関係があるのは、3D参照点の動きが経時的に相対的にスムーズであるからである。量子化投影係数のエントロピは列iの投影係数を、列i-1、つまり、$c_{i-1}+\Delta_i$から時間的に予測し、これらの列の対応する要素間の差分を符号化することによって減少することができる。このテストケースでデータ集合を圧縮するために、均一量子化を使用して係数を量子化し、その後時間的予測を使用してこれらをさらに圧縮した。以下で述べるように、他の形態の量子化を使用することも可能である。

【0234】

このデータ集合では、W_0の最初の45個の行に対応する、最初の45個の主要コンポーネントに関連する投影係数だけが顕著な時間的相関関係をもっているため、最初の45個の行だけが時間的に予測される。残りの行は直接にエントロピコード化される。時間的予測の後、エントロピこのテストケースでは、約20パーセントだけ減少している。図12は、最初の45個の係数の時間的予測が、どのようにエントロピを減少したかを示すグラフである。垂直軸はエントロピをサンプル当たりのビット数で示し、水平軸は係数のインデックスを示している。このケースでは、各係数はサンプルである。点線は予測なしの係数のエントロピをプロットしたもので、実線は予測付きの係数のエントロピをプロットしたものである。

【0235】

上述したように、基底ベクトルはこれらを量子化すると、さらに圧縮することができる。本実施形態では、基底ベクトルは、ピーク誤差レートを選択し、その後、各ベクトルに割り当てられた量子化レベルの数を各ベクトルの投影係数の標準偏差に基づいて変えることによって圧縮されている。この形態の量子化はスカラ量子化（scalar quantization – SQ）とも呼ばれている。SQは、実数を丸めによって整数に変換する量子化法である。SQでは、丸め関数（例えば、round(.)

）が次の例に示すように、実数を整数に変換している。すなわち、数4.73はround(4.73) = 5によって近似化され、数3.21はround(3.21) = 3によって近似化され、数−6.1はround(−6.1) = −6によって近似化される。なお、丸めには、近似化誤差があり、これは−0.5と0.5の範囲で変化している。つまり、その最大絶対値は0.5である。round(.)関数のとり得る値は量子化レベルとも呼ばれている。

【０２３６】

以下の例では、x_i が基底ベクトルのi 番目の座標であり、i が1からNの間で変化するものとしている。ここで、Nは、すべてのベクトルにおけるすべての座標の数である。SQ法の例は次のようになっている。

【０２３７】

1) すべてのx_i （ただし、I = 1,2,..., N) を調べ、Vをその最大絶対値と名付ける。つまり、すべてのiについて$V = \max \{ |x_i| \}$

2) 最大相対誤差（「ピーク誤差レート」）dをセットする。例えば、d = 0.001は、最大相対誤差が0.1%であることを意味する。

3) すべてのx_i を「ゲイン」係数だけスケーリングする。ただし、A = 0.5 / (d* V)。すなわち、すべてのiについて$y_i = A * x_i$ を計算する。

4) 次に、値を最寄りの整数に丸めて量子化する。つまり、すべてのi についてu_i = round(y_i)

5) 量子化した値のスケーリングを、Aの反転によって元に戻す。つまり、すべてのi について $v_i = u_i$ / Aを計算する。

【０２３８】

なお、量子化値v_i はオリジナル値x_i の近似値になっている。この近似値の質は、最大相対誤差が$\max(|x_i - v_i|) / \max \{|x_i|\} \Leftarrow d$ を満足しているので、パラメータdによって制御される。

【０２３９】

なお、v_i とu_i の間には１対１の関係がある。u_i は整数であるので、これらは有限数のビットで表されている。さらに、小さな値をもつu_i は大きい値をもつものよりもビット数が少なくなっている。

【０２４０】

最後に、画像の集合が与えられているとき、ベクトルの値u_i は確率分布が非均一になる。例えば、y_i の値の多くは非常に小さいのが代表的であるので、u_i はゼロになる。従って、量子化によると、量子化されたデータをエントロピコーダで圧縮すると、コードワードが発生確率に基づいて各値に割り当てられるので圧縮効率が向上する。図１２のグラフは、そのようなコーダのエントロピ（係数当たりの平均ビット数）を示している。

【０２４１】

別の形態の量子化を使用することも可能である。例えば、数x_i は小さなベクトルにグループ化した後で（例えば、M = 4またはM = 8の値グループ）、ベクトル量子化（vector qualification – VQ）を使用して圧縮することができる。VQでは、ベクトルはM次元空間における正規または非正規点格子内の最寄りの近隣によって近似化される。

【０２４２】

実用では、本テストケースで生成されたデータの場合、VQを使用しても向上はわずかである。多くても圧縮が20%向上するだけである。

【０２４３】

上述したように、予測係数と量子化基底ベクトルは、算術演算またはハフマン(Huffman) コード化などの、エントロピコード化を使用するとさらに圧縮される。エントロピコード化によると、発生頻度の高いサンプルには短いコードが、発生頻度の低いサンプルには長いコードが割り当てられるので、ジオメトリックデータはさらに圧縮されることになる。

【０２４４】

ピーク誤差レートをいろいろと変えてW_gを圧縮したアニメーションシーケンスをビジュアルに検査した。誤差レベルが約0.001または0.002のときのW_g のエントロピは36 Kビット/秒、U_g のエントロピは13 Kビット/秒であり、全グラフィックデータでは合計40 Kビット/秒である。これらの値は3330フレームのアニメーションシーケンス（１秒）の場合の平均である。

【０２４５】

伸張 (decompression)

ジオメトリデータは、コード化ステップを逆に実行することによって復号化される。最初に、エントロピデコーダ（復号器）は、基底ベクトルと係数を可変長コードから再構築する。次に、係数が予測係数から再構築される。次に、逆量子化器は係数と基底ベクトルを復元する。変形ベクトルのオリジナルマトリックスは、その後、基底ベクトルと係数マトリックスから再構築される。

【0246】

変形軌跡エンコーダとしてのセグメンテーションとジオメトリック変換

Guenter他によれば、システムへの入力として変形コントロールの集合が与えられている（例えば、俳優の顔またはボディ上の蛍光ドットの位置によって）。このセクションで説明している手法は、生の頂点マトリックスが与えられているとき、変形コントロールを推定している。初期頂点マトリックスVが与えられた後で、上述したセグメンテーションとジオメトリック変形手法を適用すると、V = A Bが因数分解される。ここで、Aは軌跡の集合、Bは影響重み（influence weight）の集合である。例えば、アフィンジオメトリックコーダでは、左辺は時間依存アフィン変換であり、右辺は初期ベースメッシュの集合である。

【0247】

【数36】

$$V = \{ A_0 A_1 \ldots A_{n-1} \} \, \mathrm{diag}(V_0 V_1 \ldots V_{n-1}).$$

【0248】

ジオメトリの意味では、Aは時間変化変形コントロールの集合である。アフィン例では、これらのコントロールは、オリジナル表面近くの並行移動中心周りの回転、スケール、およびスキューを定義する点の集合である。

【0249】

動作環境

プロトタイプは次の2システム上に実装されている。

1) Silicon Graphics, Inc.提供のワークステーション

a. 150MHz MIPS R4400 プロセッサおよび128 MBメモリを搭載し、IRIX 5.3 オペレーティングシステムが稼動しているIndigo2 モデル。

b. OpenGLが稼動している急進的グラフィックスコプロセッサ

2) Gateway, Inc. 提供のGateway 2000コンピュータ

a. 300MHz Intel Pentium IIプロセッサと128 MBメモリを搭載し、Microsoft Windows 98オペレーティングシステムが稼動しているE5000モデル。

b. NVIDIA RivaTNTグラフィックスチップセットを使用し、Microsoft DirectXマルチメディアAPIが稼動しているDiamond Multimedia Viper550グラフィックスカード。

【０２５０】

図１３と以下の説明は、上述したソフトウェアルーチンを実装できる適当なコンピューティング環境の概要を簡単に説明することを目的としている。

【０２５１】

図１３は、本発明用の動作環境として使用できるコンピュータシステムの例を示す図である。コンピュータシステムは従来のコンピュータ５２０を装備し、その中には、処理ユニット５２１、システムメモリ５２２、およびシステムメモリを含む種々のシステムコンポーネントを処理ユニット５２１に結合しているシステムバス５２３が含まれている。システムバスは、いくつかを挙げると、PCI、VESA、マイクロチャネル（Microchannel）、ISAおよびEISAなどの、従来の種々バスアーキテクチャのいずれかを使用する、数種タイプのバス構造のいずれかで構成することが可能であり、この中には、メモリバスまたはメモリコントローラ、周辺バス、およびローカルバスが含まれている。システムメモリとしては、リードオンリメモリ（read only memory – ROM）５２４とランダムアクセスメモリ（random access memory – RAM）５２５がある。基本入出力システム（basic input/output system – BIOS）５２６は、スタートアップ時のときのように、コンピュータ５２０内のエレメント間で情報を転送しやすくする基本ルーチンから構成され、ROM５２４に格納されている。コンピュータ５２０は、さらに、ハードディスクドライブ５２７、例えば、取り外し可能ディスク５２９に読み書きするための磁気ディスクドライブ５２８、および例えば、CD-ROMディスク５３１を読み取り、あるいは他の光媒体に読み書きするための光ディスクドライブを装備している。ハードディスクドライブ５２７、磁気ディスクドライブ５２８、および

光ディスクドライブ５３０は、それぞれハードディスクドライブインタフェース５３２、磁気ディスクドライブインタフェース５３３、および光ディスクドライブインタフェース５３４を通してシステムバス５２３に接続されている。これらのドライブおよび関連するコンピュータ可読媒体は、データ、データ構造、コンピュータ実行可能命令などをコンピュータ５２０のために保管しておく、不揮発性ストレージとなっている。上記コンピュータ可読媒体の説明では、ハードディスク、取り外し可能磁気ディスクおよびCDだけが挙げられているが、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、Bernoulliカートリッジなどのように、コンピュータが読み取れる媒体ならば、他のタイプの媒体をこのコンピューティング環境で使用することも可能である。

【０２５２】

ドライブおよびRAM５２５には、複数のプログラムモジュールを格納しておくことができる。そのようなものとしては、オペレーティングシステム、１つまたは複数のアプリケーションプログラム（上述した圧縮および伸張ルーチンなど）、他のプログラムモジュール５３７、およびプログラムデータ５３８（例えば、マトリックスV、ジオメトリック変換パラメータ、残余の精細化マトリックス、ベースメッシュなど）などがある。ユーザは、キーボード５４０およびマウス５４２などのポインティングデバイスからコマンドや情報をコンピュータ５２０に入力することができる。他の入力デバイス（図示せず）としては、マイクロフォン、ジョイスティック、ゲームパッド、サテライトディッシュ、スキャナなどがある。上記および他の入力デバイスは、システムバスに結合されているシリアルポートインタフェース５４６を通して処理ユニット５２１に接続されていることがよくあるが、パラレルポート、ゲームポート、ユニバーサルシリアルバス（universal serial bus – USB）などの、他のインタフェースで接続することも可能である。

【０２５３】

モニタ５４７または他のタイプのディスプレイデバイスも、ビデオコントローラ５４８などのインタフェースを介してシステムバス５２３に接続されている。ビデオコントローラは、ピクセル強度値を、ディスプレイ上でスキャンされるア

ナログ信号に変換することによってレンダリングパイプラインによって生成される出力画像の表示を管理している。グラフィックスワークステーションによっては、コンピュータ上の拡張スロットにプラグインされるグラフィックスアクセラレータやマザーボード上のバス構造を通してプロセッサとメモリに接続されるグラフィックスレンダリングチップセットのような、追加のレンダリングデバイスを装備しているものもある。この種のグラフィックスレンダリングハードウェアは、代表例として、専用ハードウェアを使用して、Vにメッシュで形成されたポリゴン（多角形）などの、ジオメトリックプリミティブをスキャン変換することによって画像生成を加速化している。

【０２５４】

コンピュータ５２０は、リモートコントローラ５４９などの、１つまたは複数のリモートコンピュータとの論理的コネクションを使用するネットワーキング環境で動作させることができる。リモートコンピュータ５４９はサーバ、ルータ、ピアデバイスまたは他の共通ネットワークノードにすることができ、コンピュータ５２０に関連して上述したエレメントの多くまたはすべてを含んでいるのが代表的である。なお、図１３には、メモリストレージデバイス５５０だけが示されている。図１３に示す論理的コネクションとしては、ローカルエリアネットワーク（local area network – LAN）５５１と広域ネットワーク(wide area network – WAN) ５５２がある。このようなネットワーキング環境は、オフィス、エンタープライズワイドコンピュータネットワーク、イントラネットおよびインターネット（the Internet）で普及している。

【０２５５】

LANネットワーキング環境で使用されるときは、コンピュータ５２０はネットワークインタフェースまたはアダプタ５５３を通してローカルネットワーク５５１に接続されている。WANネットワーキング環境で使用されるときは、コンピュータ５２０は、インターネットなどの広域ネットワーク５５２上で通信を確立するためのモデム５５４または他の手段を装備しているのが代表的である。モデム５５４は、外部モデムと内部モデムのどちらも、シリアルポートインタフェース５４６を介してシステムバス５２３に接続されている。ネットワーキング環境で

は、コンピュータ５２０に関連して示されているプログラムモジュールまたはその一部は、リモートメモリストレージデバイスに格納しておくことができる。図示のネットワークコネクションは単なる例であり、コンピュータ間の通信リンクを確立する他の手段を使用することも可能である。

【０２５６】

【表６】

圧縮結果の例

アフィン量子化	頂点量子化	ＳＮＲ	量子化サイズ	圧縮比	サイズ	圧縮比,Gzip	減衰帯域幅
(bits)	(bits)	(dB)	(bytes)		(bytes)		(Kbytes/s)
10	4	27.69	1962000	7.4	433734	33	63.5
12	4	31.33	1990800	7.3	467153	31	68.4
14	4	32.06	2019600	7.2	510058	28	74.7
16	4	32.11	2048400	7.1	526982	27	77.2
10	8	28.00	3780000	3.8	1004086	14	147.1
12	8	33.90	3808800	3.8	1036927	14	151.9
14	8	39.77	3837600	3.8	1079085	13	158.1
16	8	43.37	3866400	3.8	1096496	13	160.6
NA	16	46.93	7272000	2.0	6654007	2.2	974.7

表 1

【０２５７】

表１は、コーデックでの結果であり、そこでは、残余行予測によるジオメトリック変換コード化（Affin-Match）によって時間依存頂点マトリックスが圧縮され、その後に続いてエントロピコード化（GNUのgzipのLempel-Ziv手法を使用）が行われている。この表は、未圧縮浮動小数点頂点マトリックス（サイズ14544000バイト＝400フレーム＊3030頂点/フレーム＊3座標/頂点＊4バイト/座標）と圧縮メッシュとの比率別にソートされている。比較のために、最後の行には、16ビットに量子化された生の（raw）頂点マトリックスが示されている。

【０２５８】

【表７】

	Chicken				顔			
	サイズ (Mbytes)	比率	スタートアップ (Kbytes)	帯域幅 (Kbyte/s)	サイズ (Mbytes)	比率	スタートアップ (Kbytes)	帯域幅 (Kbyte/s)
未圧縮	13.9	1	66.4	2,130.5	21.7	1	36.8	622.3
MPEG4 (static)	2.7	5	1.6	409.3	3.8	6	0.6	107.8
Guenter et al	-	-	-	-	1.2	19	～1000	5.0
アフィン合致	0.5	27	57.9	68.6	3.9	6	51.3	109.7
アフィン合致（顔ドット軌跡に適用）					0.7	33	74.4	16.7

表 2

【０２５９】

表２は、各種圧縮手法の比較を示している。これらの結果から分かるように、アニメーション化モデルがアフィン変換モデル（Chicken欄の最下段行）に近いとき、アフィンマッチ手法が高圧縮になっている。MPEG4の実験用圧縮器は階層のアニメーションを活用する設計になっていない。顔の例では、全頂点マトリックスに適用される汎用アフィン合致圧縮器は、各フレームに適用されるMPEG4実験用コーダよりも性能が悪くなっている。ドット軌跡に適用されたときは、アフィン合致圧縮器は良好な働きをする。このことから分かるように、圧縮器は、最も適合するエンコーダを見つけるためには、アフィン合致だけよりも、大きなクラスのアニメーションモデルを探さなければならない。

【０２６０】

代替実施形態

以上、可能とされるいくつかの実施形態を参照して本発明を説明してきたが、本発明はこれらの特定実施形態の範囲に限定されないことはもちろんである。本発明の実施形態には、上述してきた新規コード化方法の１つまたは２つ以上を取り入れることが可能である。さらに、上述したコード化方法は、他のコード化方法と併用することが可能である。時間変化ジオメトリデータを表すマトリックス

には、離散コサイン変換（Discrete Cosine Transform — DCT）、ウェーブレット/ゼロツリーコーダのような、周波数ドメインコーダを適用することが可能であり、この中には、頂点位置のマトリックス、変形ベクトルや奇跡のマトリックス、および残余のマトリックスが含まれている。これらのコード化方式は、上述したコード化方法で残余マトリックスに適用したとき特に効果的である。ジオメトリック構造が除かれた後（例えば、アフィン変換ベースメッシュがカレントの頂点位置マトリックスから除かれた後）、周波数ドメインコーダを残余マトリックスに適用することができる。周波数ドメインコーダに加えて、エントロピコーダと量子化を使用すると、変化するジオメトリストリーム内のデータをさらに圧縮することができる。

【０２６１】

圧縮手法は、上述した空間と時間だけでなく、他の次元にわたっても適用することが可能である。

【０２６２】

本発明の原理を適用できる実施形態は多数が可能であるので、当然に理解されるように、上述してきた実施形態は本発明の単なる例であり、本発明の範囲を限定するものではない。むしろ、本発明の範囲は特許請求の範囲に定義されている通りである。従って、特許請求の範囲に記載されている特徴はすべて、本発明の範囲と精神に属するものである。

【図面の詳細な説明】

【図１】

メッシュ変換コーダの例を示すブロック図である。

【図２】

変換マッチングを実行して変換パラメータを時間依存ジオメトリから導き出すベースメッシュ変換コーダを示すブロック図である。

【図３】

ジオメトリック変換ではなく、逆ジオメトリック変換を符号化する代替ベースメッシュ変換コーダを示すブロック図である。

【図４】

先行タイムサンプル用に計算されたメッシュを、カレントタイムサンプルのベースメッシュとして使用する別タイプのメッシュ変換コーダを示すブロック図である。

【図5】

マトリックス予測コーダの例を示すブロック図である。

【図6】

主要コンポーネント分析を使用して時間依存ジオメトリストリームを圧縮するメッシュ基底コーダを示すブロック図である。

【図7】

回転しながら、並行移動する立方体のアニメーションシーケンスを示す図であり、そこでは、アニメーションのいくつかの側面が別々のフレーム列に示されている。つまり、1) 最下行は左から右へ単純に並行移動する立方体のジオメトリを示し、2) 中間行は立方体の上部が立方体の下部よりも高速に回転しているが、立方体の辺が直線のままになっている中間形態のアニメーションを示し、3) 最上行は立方体の曲線の動きが複雑化し、その頂点が立方体の下部から上部に徐々に高速に回転していることを示している。

【図8】

空間と時間の両方における詳細レベル間の空間－時間ピラミッドの構造を示す図である。

【図9】

メッシュ変換コード化、基底コード化、メッシュ単純化および行/列予測コード化を結合した時間依存ジオメトリのプロトタイプ圧縮器を示すブロック図である。

【図10】

図9の圧縮器によって出力された圧縮時間依存ジオメトリの伸張器を示すブロック図である。

【図11】

主要コンポーネント分析、時間的予測、量子化およびエントロピコード化を使用して時間変化ジオメトリを符号化するコーダを示すブロック図である。

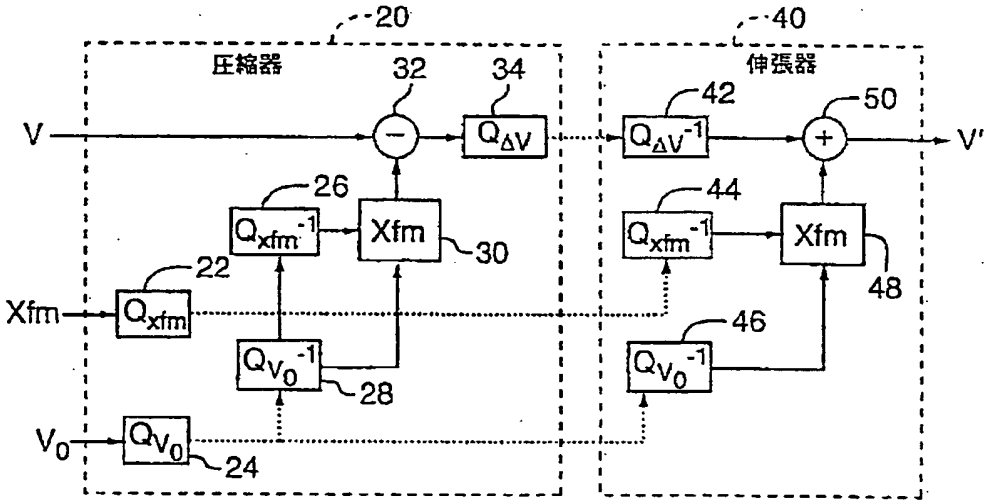
【図１２】

変形ベクトルの主要コンポーネント分析によって生成された係数の時間的予測が係数のエントロピを低減化する様子を示すグラフである。

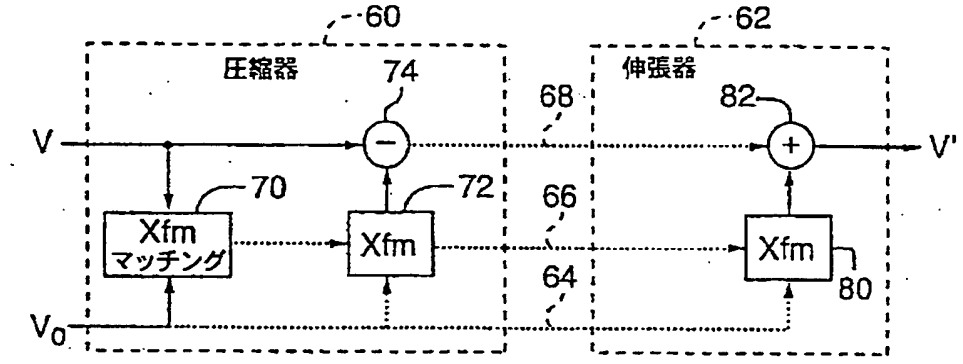
【図１３】

本発明をソフトウェアで実現したときの動作環境の働きをするコンピュータシステムを示すブロック図である。

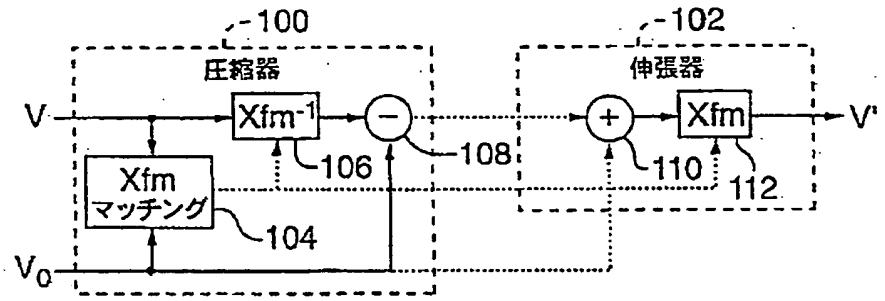
【図１】



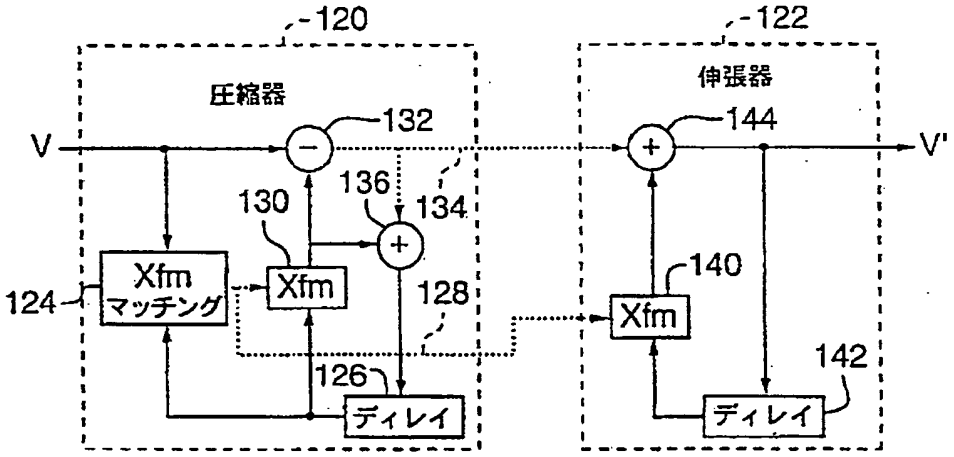
【図２】



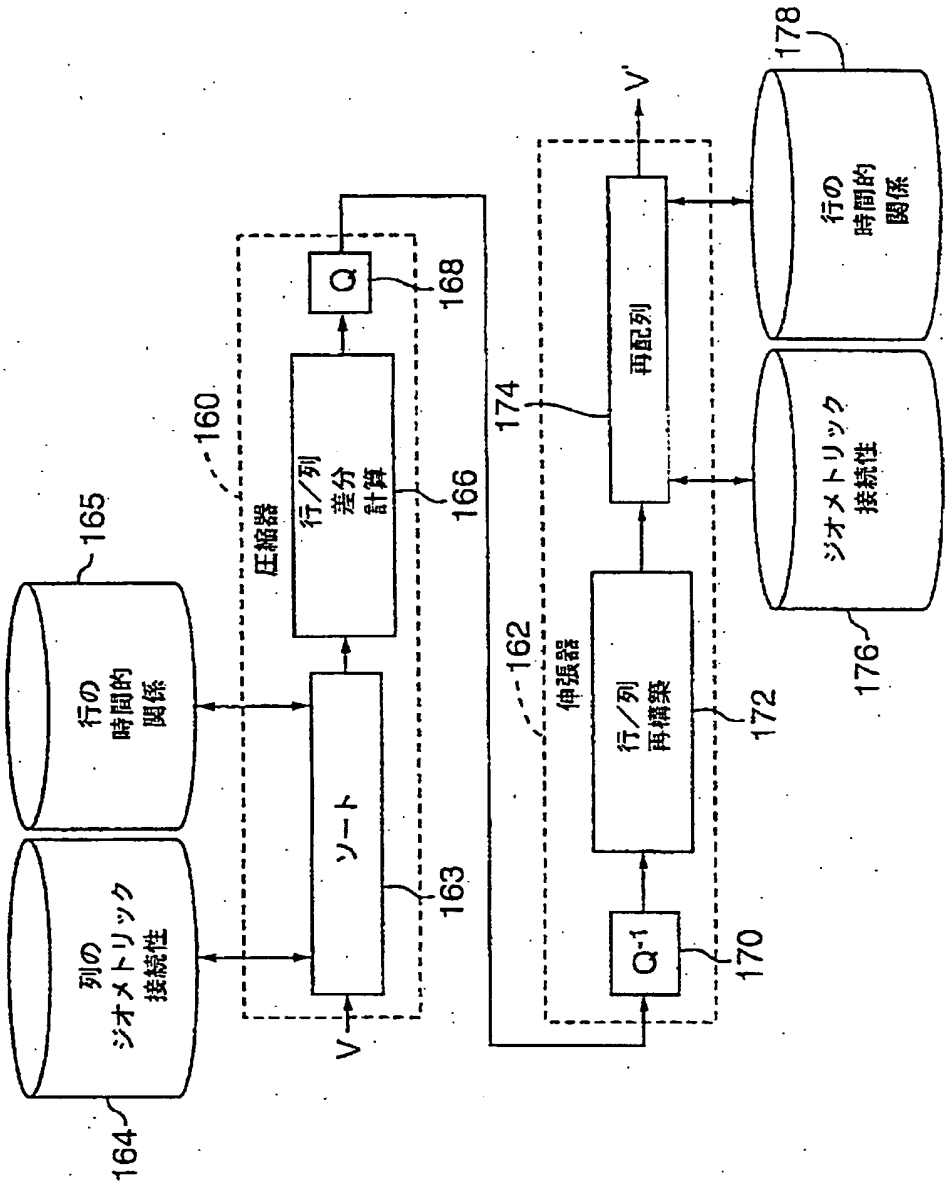
【図３】


100
102
圧縮器
伸張器
V
Xfm^{-1}
Xfm
V'
106
108
110
112
Xfm
マッチング
104
V_0

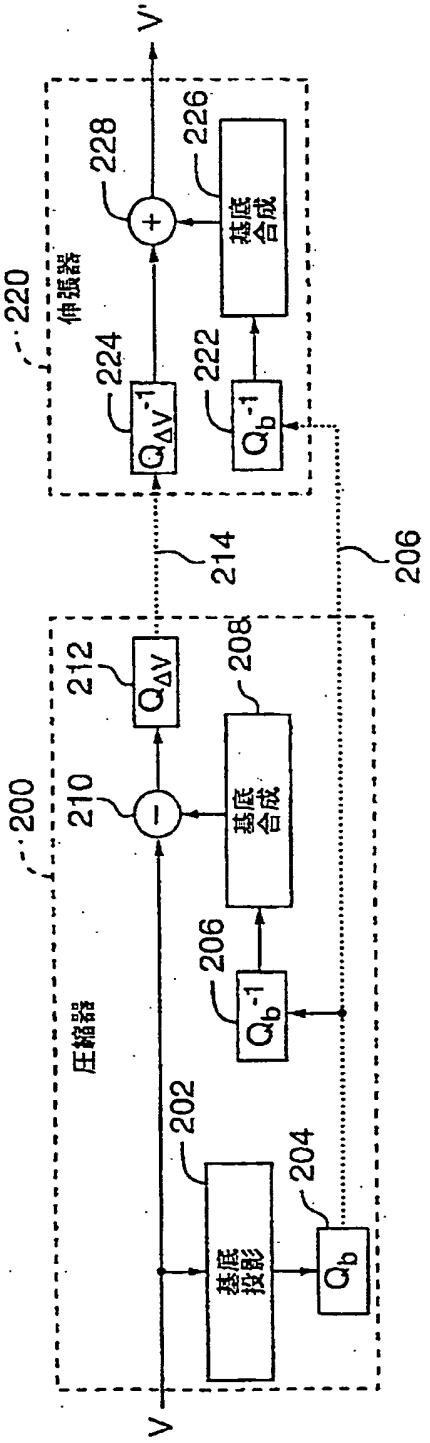
【図４】


120
122
圧縮器
伸張器
V
132
144
V'
136
134
130
140
Xfm
マッチング
124
Xfm
128
Xfm
142
126
ディレイ
ディレイ

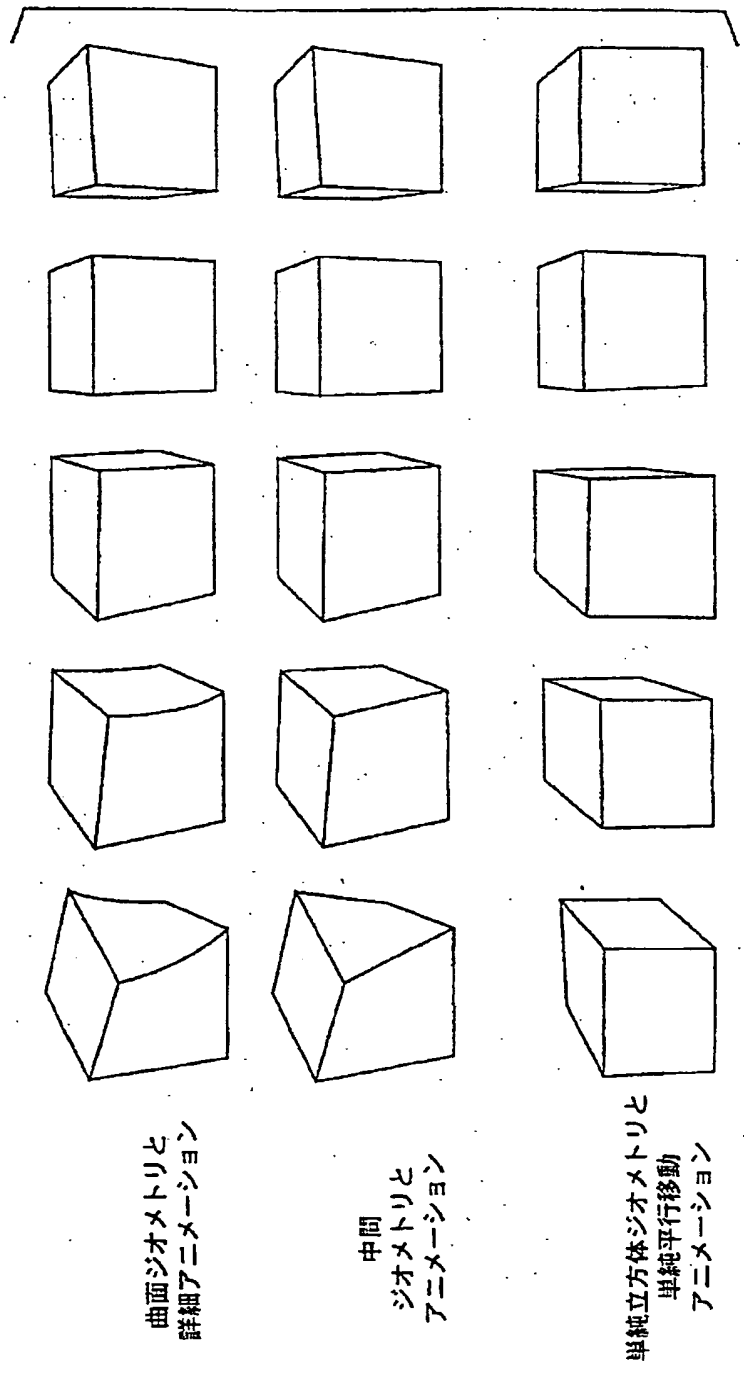
【図5】


列の
シオメトリック
接続性
164
行の
時間的
関係
165
圧縮器
160
ソート
163
行／列
差分
計算
166
Q
168
V
伸張器
162
行／列
再構築
172
Q-1
170
再配列
174
V'
行の
時間的
関係
178
シオメトリック
接続性
176

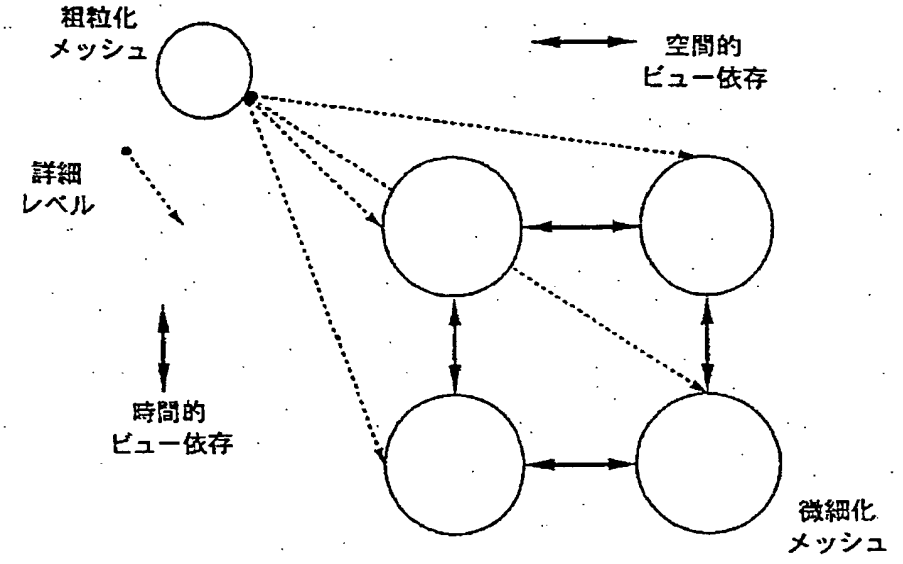
【図6】


V'
220
伸張器
228
226
基底合成
224
222
$Q_{\Delta V}^{-1}$
Q_b^{-1}
214
206
200
圧縮器
210
212
$Q_{\Delta V}$
208
基底合成
206
Q_b^{-1}
202
基底投影
204
Q_b
V

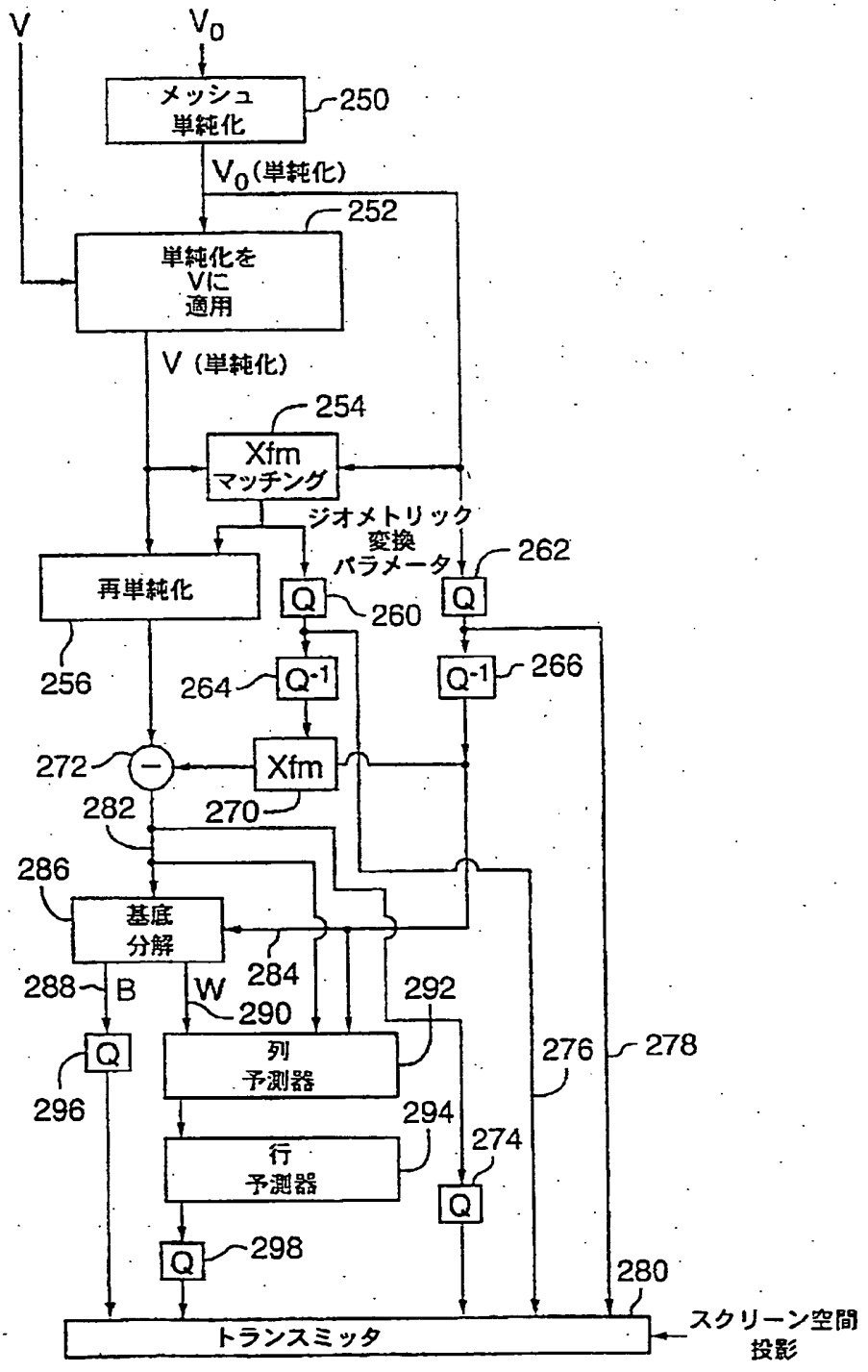
【図７】


曲面ジオメトリと
詳細アニメーション
中間
ジオメトリと
アニメーション
単純立方体ジオメトリと
単純平行移動
アニメーション

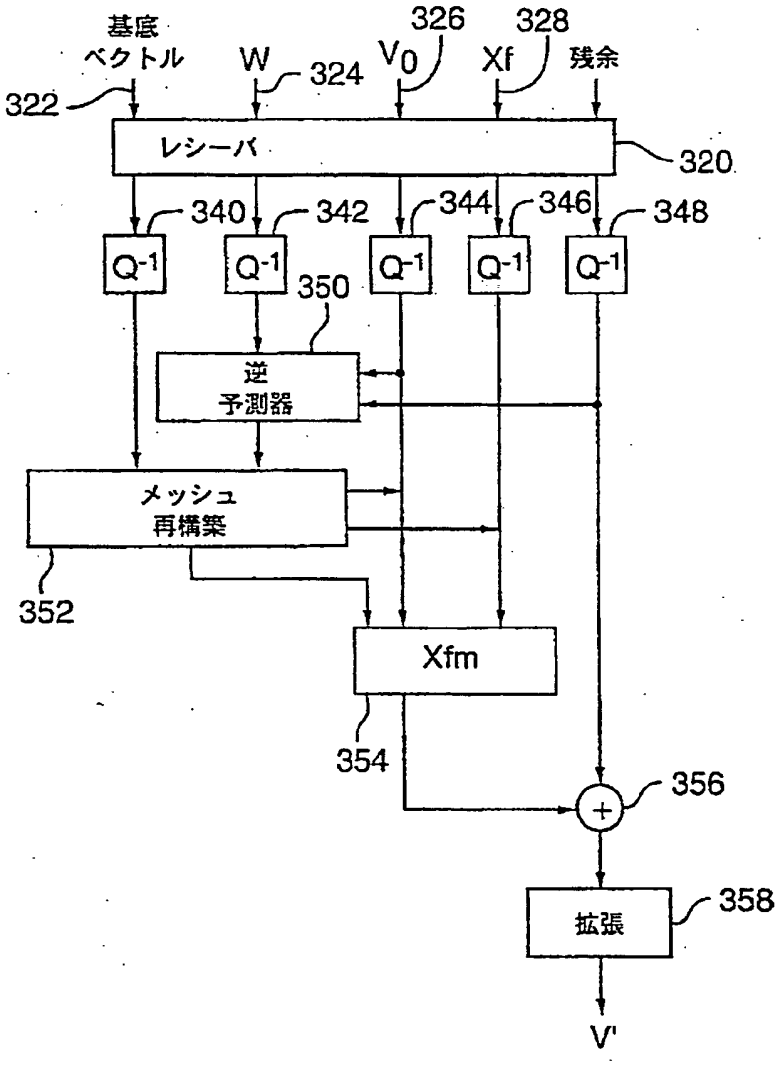
【図８】



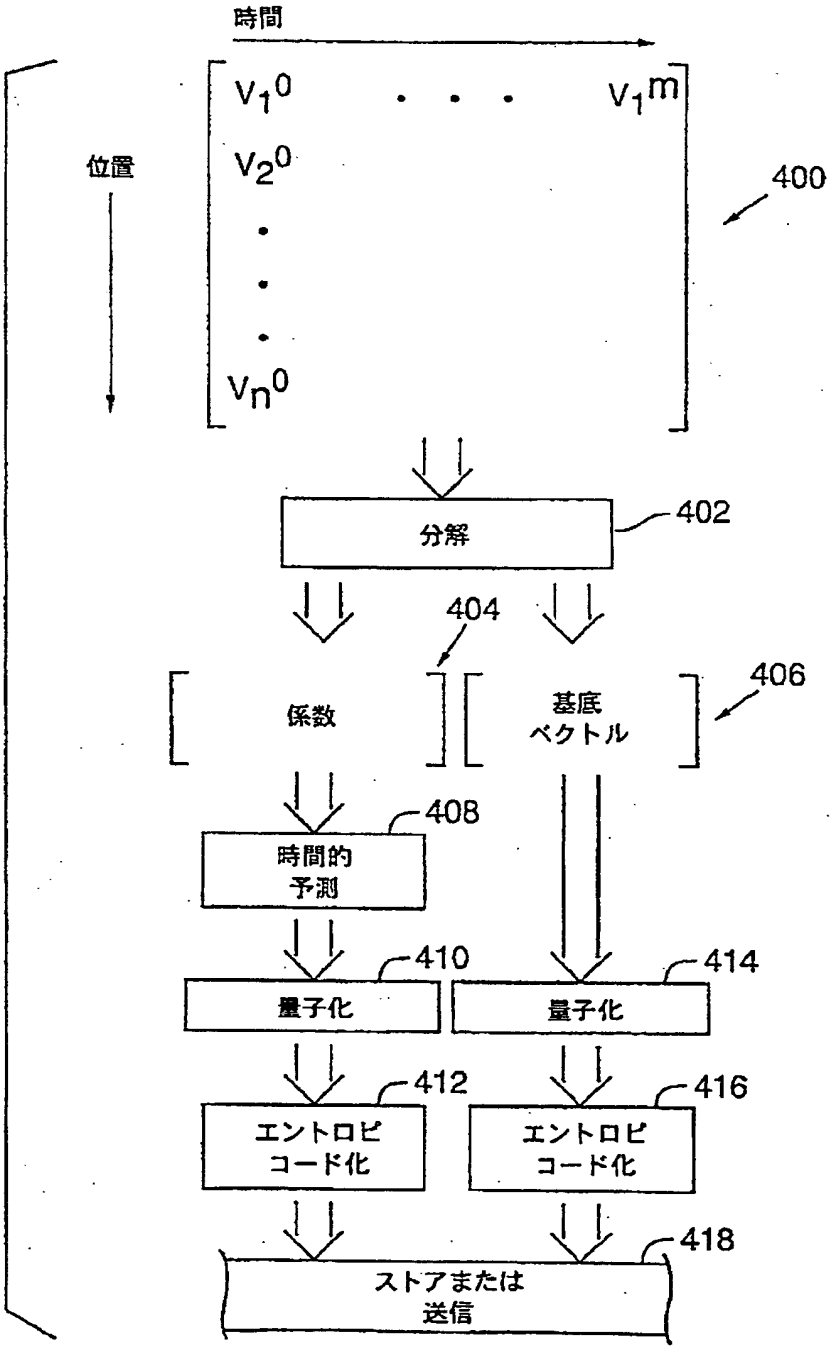
【図9】


V
V_0
メッシュ単純化
250
V_0(単純化)
252
単純化をVに適用
V (単純化)
254
Xfm マッチング
ジオメトリック変換パラメータ
262
再単純化
Q
260
Q
256
264
Q^{-1}
Q^{-1}
266
272
Xfm
270
282
286
基底分解
284
288
B
W
290
292
Q
列予測器
276
278
296
294
274
行予測器
Q
Q
298
280
トランスミッタ
スクリーン空間投影

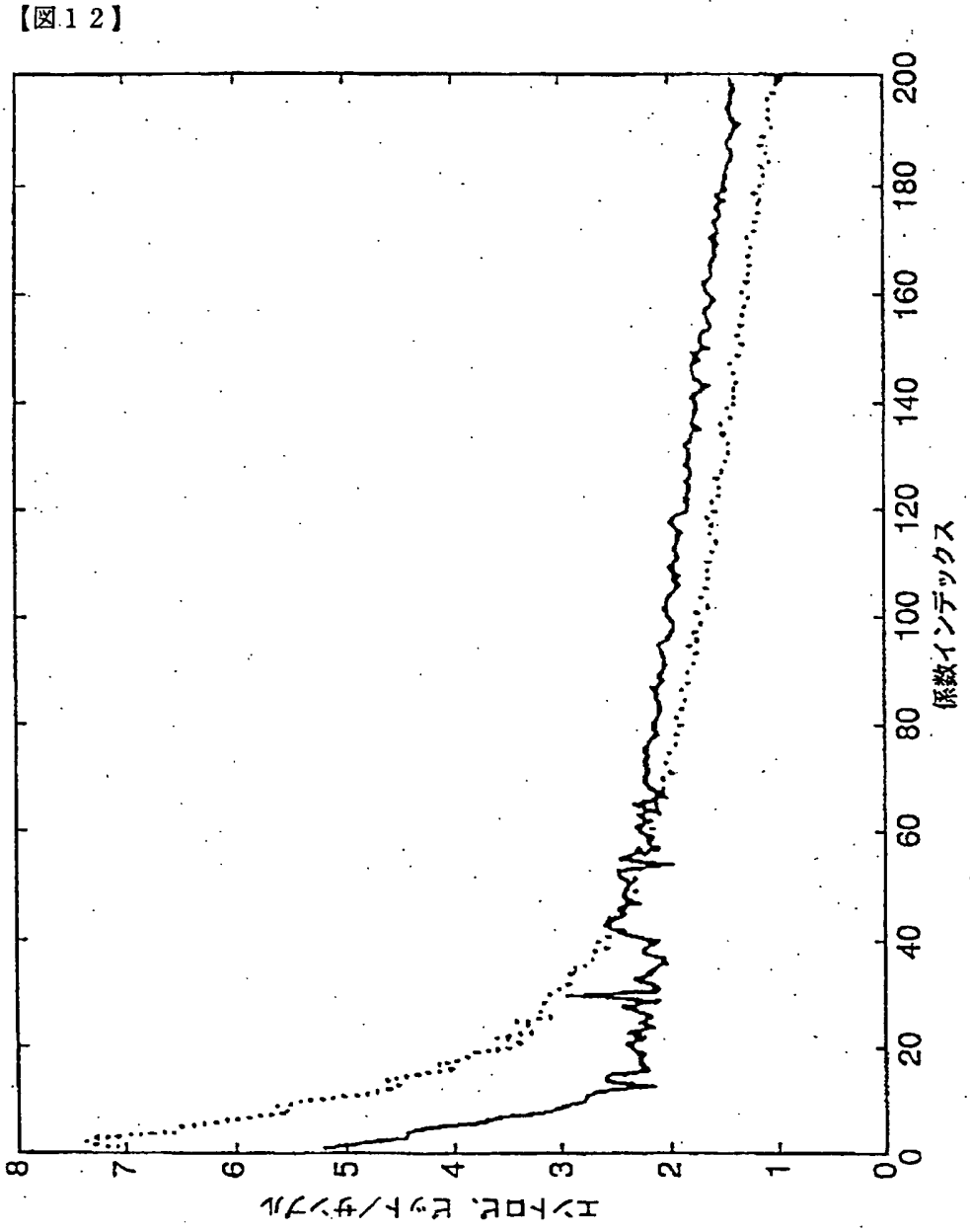
【図１０】


基底
ベクトル
322
W
324
326
V0
328
Xf
残余
レシーバ
320
340
342
344
346
348
Q-1
Q-1
Q-1
Q-1
Q-1
350
逆
予測器
メッシュ
再構築
352
Xfm
354
356
拡張
358
V'

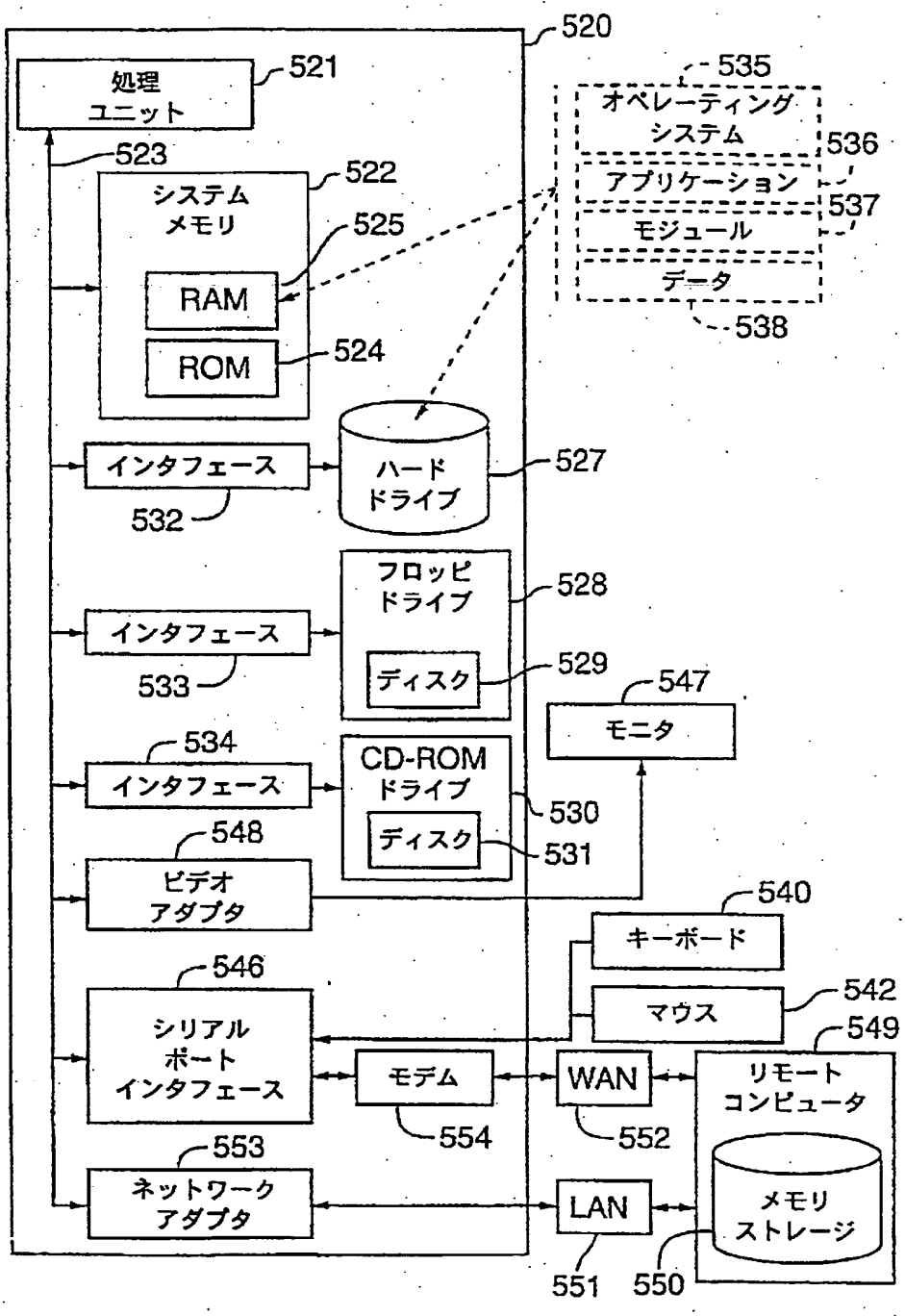
【図１１】


時間
位置
V_1^0 ・・・ V_1^m
V_2^0
V_n^0
400
分解
402
404
係数
基底ベクトル
406
時間的予測
408
量子化
410
量子化
414
エントロピコード化
412
エントロピコード化
416
ストアまたは送信
418

【図12】


ハイドロゲン ピットサンプル
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100
120
140
160
180
200
係数インデックス

【図１３】


520
処理ユニット
521
535
オペレーティングシステム
536
アプリケーション
537
モジュール
データ
538
523
システムメモリ
522
525
RAM
ROM
524
インタフェース
532
ハードドライブ
527
フロッピドライブ
528
インタフェース
533
ディスク
529
547
モニタ
534
インタフェース
CD-ROMドライブ
530
ディスク
531
548
ビデオアダプタ
540
キーボード
546
マウス
542
549
シリアルポートインタフェース
モデム
WAN
リモートコンピュータ
554
552
553
ネットワークアダプタ
LAN
メモリストレージ
551
550

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/12732

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06T 17/00
US CL :345/418, 419, 423
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 345/418, 419, 420, 423, 424, 427, 428, 429, 473, 474

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,729,671 A (PETERSON et al) 17 March 1998	1-55
A	US 5,736,991 A (TADA) 07 April 1998	1-55
A	US 5,751,931 A (COX et al) 12 May 1998	1-55
A,E	US 5,914,721 A (LIM) 22 June 1999	1-55
A,E	US 5,945,996 A (MIGDAL et al) 31 August 1999	1-55

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 SEPTEMBER 1999	07 FEB 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer CLIFF NGUYEN VO Telephone No. (703) 305-9594

Form PCT/ISA/210 (second sheet)(July 1992)*

フロントページの続き

(81)指定国 EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG), AP(GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), EA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW

(72)発明者 ブライアン ジェンター
アメリカ合衆国 98052 ワシントン州 レッドモンド ノースイースト 41 ストリート 16725

(72)発明者 ヘンリク サーメント マルバー
アメリカ合衆国 98053 ワシントン州 レッドモンド 23 アベニュー ノースイースト 2302

Fターム(参考) 5B050 AA09 BA08 BA09 BA18 CA07 EA02 EA10 EA12 EA24 EA27 EA28 FA02
5B080 AA12 BA05 BA08 DA06 FA15 GA25
5C059 LB11 MA00 MA01 MA24 MA31 MD02 PP12 PP13 SS20 SS26 UA02 UA05 UA31

【要約の続き】
ので、この形態のコード化が最適化される。空間的－時間的詳細レベルコーダは、時間依存ジオメトリストリームを階層構造に変換し、そこには空間次元と時間次元の詳細レベルと拡張レコードが含まれている。拡張レコードは詳細レベル間の差分を表すデルタからメッシュをどのように再構築するかを指定している。低帯域幅伝送のために、カレントタイムサンプルのとき表現されているオブジェクトがどの詳細レベルにあるかに応じて、選択した拡張レコードだけを送信すればよいようになっている。